Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Uso de modelo de aprendizado de máquina em microcontroladores para detecção preventiva de danos e falhas em máquinas e equipamentos

João Guilherme Mendanha Alves

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Luiz Carlos Bambirra Torres

COORIENTAÇÃO:
Alexandre Magno de Sousa

Fevereiro, 2026
João Monlevade–MG

João Guilherme Mendanha Alves

Uso de modelo de aprendizado de máquina em microcontroladores para detecção preventiva de danos e falhas em máquinas e equipamentos

Orientador: Luiz Carlos Bambirra Torres

Coorientador: Alexandre Magno de Sousa

Monografia apresentada ao curso de Sistemas de Informação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Fevereiro de 2026

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

A474u Alves, Joao Guilherme Mendanha.
Uso de modelo de aprendizado de máquina em microcontroladores para detecção preventiva de danos e falhas em máquinas e equipamentos. [manuscrito] / Joao Guilherme Mendanha Alves. - 2026.
65 f.: il.: color., gráf., tab..

Orientador: Prof. Dr. Luiz Carlos Bambirra Torres.
Coorientador: Prof. Dr. Alexandre Magno de Souza.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Graduação em Sistemas de Informação .

1. Aprendizado do computador. 2. Industria 4.0. 3. Inteligência artificial. 4. Máquinas - Indústria - Medidas preventivas. 5. Microcontroladores. 6. Sistemas embarcados (Computadores). I. Torres, Luiz Carlos Bambirra. II. Souza, Alexandre Magno de. III. Universidade Federal de Ouro Preto. IV. Título.

CDU 658.58:004.85

Bibliotecário(a) Responsável: Flavia Reis - CRB6/2431

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

João Guilherme Mendanha Alves

Uso de modelo de aprendizado de máquina em microcontroladores para detecção preventiva de danos e falhas em máquinas e equipamentos

Monografia apresentada ao Curso de Sistemas de Informação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação

Aprovada em 05 de março de 2026

Membros da banca

Doutor Luiz Carlos Bambirra Torres - Orientador (Universidade Federal de Ouro Preto)
Doutor Alexandre Magno de Sousa - Coorientador (Universidade Federal de Ouro Preto)
Doutora Janniele Aparecida Soares Araújo - (Universidade Federal de Ouro Preto)
Doutor Roberto Gomes Ribeiro - (Universidade Federal de Ouro Preto)

Luiz Carlos Bambirra Torres, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 24/03/2026

Documento assinado eletronicamente por **Luiz Carlos Bambirra Torres**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 24/03/2026, às 20:39, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **1080824** e o código CRC **C7EDA2FE**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: (31)3808-0819 - www.ufop.br

Este trabalho é dedicado a todo o esforço que me trouxe até aqui.

Agradecimentos

Agradeço a Deus e à minha família por terem sido meu suporte nessa fase tão importante da minha vida. Deus foi meu amparo nos momentos de incerteza, minha força quando o cansaço parecia maior que a vontade, e minha luz quando o caminho parecia escuro.

À minha família, deixo minha gratidão eterna. Obrigado por cada palavra de incentivo, por cada gesto de carinho, pelas noites em que se preocuparam comigo e pelos dias em que comemoraram minhas pequenas conquistas como se fossem grandes vitórias. Vocês acreditaram em mim quando até eu duvidava e me ensinaram que perseverar vale a pena.

Essa conquista não é apenas minha, é de todos nós. É o resultado da fé que me guiou, do amor que me sustentou e do apoio que me deu forças para seguir firme até o fim. Carrego comigo tudo o que aprendi e o orgulho de saber que cheguei até aqui porque nunca caminhei sozinho.

"A mente que se abre a uma nova ideia, jamais voltará ao seu tamanho original."

— Albert Einstein.

Resumo

A manutenção preditiva é um pilar fundamental para a eficiência e competitividade na Indústria 4.0, porém, seu custo de implementação em larga escala ainda é um desafio, especialmente para pequenas e médias empresas. Este trabalho propõe o desenvolvimento e a validação de um protótipo de baixo custo para detecção preventiva de falhas em equipamentos industriais, utilizando aprendizado de máquina diretamente em um microcontrolador. A metodologia fundamenta-se no paradigma TinyML, empregando um microcontrolador ESP32 e o classificador Naive Bayes para realizar inferências localmente. O modelo foi treinado e validado utilizando um conjunto de dados sintético do domínio industrial, submetido a etapas de pré-processamento, seleção de características e balanceamento. Os resultados obtidos a partir de 30 máquinas distintas demonstraram alta eficácia, com acurácia entre 91% e 96%, precisão de até 96% e recall de até 98%. A aplicação de um limiar de decisão probabilístico de 80% mostrou-se eficaz para reduzir falsos positivos sem comprometer a sensibilidade do sistema. Conclui-se que a integração de hardware de baixo custo com algoritmos de aprendizado de máquina otimizados é uma alternativa tecnicamente viável e financeiramente acessível, capaz de democratizar o acesso à manutenção preditiva e contribuir para a redução de paradas não planejadas e custos operacionais na indústria nacional.

Palavras-chaves: TinyML. Manutenção Preditiva. ESP32. Naive Bayes. Indústria 4.0. Aprendizado de Máquina.

Abstract

Predictive maintenance is a fundamental pillar for efficiency and competitiveness in Industry 4.0; however, the cost of its large-scale implementation remains a challenge, especially for small and medium-sized enterprises. This work proposes the development and validation of a low-cost prototype for the preventive detection of failures in industrial equipment, using machine learning directly on a microcontroller. The methodology is based on the TinyML paradigm, employing an ESP32 microcontroller and the Naive Bayes classifier to perform inferences locally. The model was trained and validated using a synthetic dataset from the industrial domain, which underwent preprocessing steps, feature selection, and balancing. The results obtained from 30 different machines demonstrated high effectiveness, with accuracy ranging from 91% to 96%, precision up to 96%, and recall up to 98%. The application of a probabilistic decision threshold of 80% proved effective in reducing false positives without compromising system sensitivity. It is concluded that the integration of low-cost hardware with optimized machine learning algorithms is a technically viable and financially accessible alternative, capable of democratizing access to predictive maintenance and contributing to the reduction of unplanned downtime and operational costs in the national industry.

Key-words: TinyML. Predictive Maintenance. ESP32. Naive Bayes. Industry 4.0. Machine Learning.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

IA Artificial Intelligence (Inteligência Artificial)

ML Machine Learning (Aprendizado de Máquinas)

IoT Internet of Things (Internet das Coisas)

PdM Predictive Maintenance (Manutenção Preditiva)

CPS Cyber-Physical Systems (Sistemas Ciber-Físicos)

TinyML Tiny Machine Learning

TI Tecnologia da Informação

PCA Principal Component Analysis (Análise de Componentes Principais)

UCP Unidade Central de Processamento

RAM Random Access Memory (Memória de Acesso Aleatório)

MCU Microcontrolador

ROM/Flash Read-Only Memory (Memória Somente de Leitura)

RM Reactive Maintenance (Manutenção Reativa)

CLP Controlador Lógico Programável

SVM Suporte Vector Machine

KNN K-Nearest Neighbors

IIoT Industrial Internet of Things (Internet das Coisas Industrial)

IA Inteligência Artificial

Lista de símbolos

$\sum$	Somatório
$\prod$	Produtório
$\sqrt{\cdot}$	Raiz quadrada
π	Pi
exp	Função exponencial
ln	Logaritmo natural
μ	Média
σ	Desvio padrão
λ	Autovalor
Θ	Vetor aleatório
α	Hiperparâmetro
$\gg$	Muito maior que
$\subset$	Subconjunto
$\mathbf{x}$	Vetor de características
$\hat{y}$	Valor estimado/predito

Sumário

1 Introdução

A evolução dos processos industriais, consolidada sob o paradigma da Indústria 4.0, redefiniu a forma como ativos físicos são geridos, priorizando a integração entre sistemas físicos e digitais. Nesse cenário, a gestão de ativos evoluiu de práticas puramente reativas para estratégias de *Predictive Maintenance (Manutenção Preditiva) (PdM)*. Segundo Mobley (2002), a PdM fundamenta-se no monitoramento direto da condição operacional, o que permite intervenções baseadas no estado real do equipamento, em vez de cronogramas fixos.

Essa transição tecnológica é sustentada pelo conceito de *Cyber-Physical Systems (Sistemas Ciber-Físicos) (CPS)*, que utilizam redes de sensores para coletar dados em tempo real, permitindo que algoritmos identifiquem padrões de degradação como dito por Lee, Bagheri e Kao (2014). Monostori (2014) destaca que a detecção precoce de falhas é o pilar fundamental para evitar a quebra catastrófica, utilizando ferramentas inteligentes para mapear a saúde do ativo.

Contudo, de acordo com Ray (2021), o processamento desses dados tradicionalmente dependia da computação em nuvem (*Cloud Computing*), o que gerava desafios de latência e altos custos de infraestrutura. Como alternativa viável, a Computação de Borda (*Edge Computing*) surge como uma solução para processar informações localmente, garantindo respostas rápidas e maior segurança de dados na ponta da operação.

Apesar dos benefícios comprovados, a implementação de sistemas preditivos sofisticados em parques industriais inteiros é frequentemente limitada pelo alto investimento inicial em sensores proprietários e servidores. A contribuição central deste trabalho consiste na aplicação do conceito de *Tiny Machine Learning (TinyML)*, que, segundo Warden e Situnayake (2019), é uma área da computação de borda que permite a execução de modelos de *Machine Learning (Aprendizado de Máquinas) (ML)* em dispositivos com recursos computacionais limitados, como microcontroladores de baixo custo.

O objetivo deste trabalho é a construção de um protótipo fundamentado em hardware acessível como a família de microcontroladores ESP32, capaz de realizar a detecção preventiva de falhas e danos localmente. Conforme defendido por Dutta e Bharali (2020), a eficiência do TinyML permite que a inteligência seja distribuída diretamente nos ativos, reduzindo drasticamente a necessidade de infraestruturas de Tecnologia da Informação (TI) complexas e caras.

Esta abordagem foca na democratização da tecnologia através da relação custo-benefício. Ao utilizar componentes de baixo valor comercial para realizar inferências de ML em tempo real, o projeto viabiliza o monitoramento de máquinas que antes eram

negligenciadas por não justificarem o custo de sistemas industriais como descrito por Warden e Situnayake (2019).

A relevância desta proposta é corroborada por dados da CNI (2018 - 2022), que apontam a manutenção ineficiente e a obsolescência de maquinário como grandes gargalos da produtividade brasileira. Relatórios da ABRAMAN (2021) revelam que o custo da manutenção realizada após a falha pode ser até três vezes superior ao da manutenção preditiva.

Além disso, a falta de tecnologias acessíveis impede que uma parcela significativa das indústrias nacionais implemente monitoramento automatizado, resultando em paradas não planejadas que reduzem a capacidade produtiva em até 20% em setores críticos como demonstrado pela CNI (2018 - 2022). Este trabalho contribui para mitigar esses prejuízos, oferecendo uma solução técnica eficiente e financeiramente viável para a detecção preventiva de danos, alinhando a teoria acadêmica às necessidades reais do mercado nacional.

1.1 Justificativa e motivação

A justificativa deste trabalho fundamenta-se na necessidade crítica de mitigar as perdas produtivas e financeiras na indústria nacional, as quais são frequentemente agravadas por estratégias de manutenção puramente reativas. Conforme indicam os dados da ABRAMAN (2021), o custo de reparo após a falha é drasticamente superior ao monitoramento preditivo, tornando a eficiência operacional um desafio constante.

Neste cenário, a motivação principal reside na democratização do acesso a tecnologias de ponta, como o ML, através do uso de microcontroladores de baixo custo e TinyML. Ao transpor as barreiras econômicas que impedem pequenas e médias empresas de adotarem soluções proprietárias caras, como apresentado por CNI (2018 - 2022), este projeto busca validar uma alternativa técnica que processa dados na borda (*edge*) com alta eficiência energética e precisão. Assim, a motivação deste estudo é provar que a inteligência artificial embarcada pode ser uma ferramenta acessível para a detecção preventiva de falhas, transformando a realidade técnica e econômica da industria brasileira.

1.2 Definição do problema

O principal desafio abordado neste trabalho reside na heterogeneidade do ambiente industrial, onde uma parcela significativa dos ativos não opera conectada a sistemas computacionais robustos ou de alta capacidade de processamento. Muitas máquinas operam em condições adversas, locais remotos ou linhas de produção onde a conectividade à rede é instável, limitada ou inexistente. Essa realidade técnica torna inviável a dependência

exclusiva de infraestruturas de nuvem para o processamento de grandes volumes de dados e a execução de modelos de ML voltados à PdM descrito por Ray (2021) e Mobley (2002).

Para superar essas limitações de poder computacional e conectividade, este projeto propõe a implementação de modelos de ML diretamente em microcontroladores, fundamentando-se no paradigma do TinyML descrito por Warden e Situnayake (2019). Microcontroladores representam uma alternativa viável, de baixo custo e escalável para a coleta e o tratamento contínuo de informações na ponta (*edge*) embasado na obra de Dutta e Bharali (2020).

Ao otimizar os modelos de ML para que a inferência ocorra localmente, a abordagem proposta permite a detecção precoce de anomalias sem a necessidade de envio constante de dados para servidores externos. Essa arquitetura embarcada oferece benefícios estratégicos como reduzir drasticamente a latência e garantir a confiabilidade operacional mesmo em ambientes isolados. Por fim, o objetivo é viabilizar sistemas que contribuam para a extensão da vida útil dos equipamentos e a redução dos custos operacionais, atacando os problemas de produtividade identificados no cenário nacional por ABRAMAN (2021), CNI (2018 - 2022).

1.3 Objetivos geral e específico

O objetivo geral deste trabalho consiste em desenvolver e validar um protótipo de sistema embarcado de baixo custo, fundamentado em técnicas de TinyML, para a detecção preditiva de falhas e anomalias em ativos industriais. Busca-se a implementação de um modelo de ML otimizado para execução local em microcontroladores, garantindo autonomia, baixa latência e independência de infraestruturas de conectividade robustas. Para alcançar o objetivo geral, definem-se os seguintes objetivos específicos:

- Avaliar a viabilidade técnica de microcontroladores de baixo custo (como o ESP32) para o processamento de modelos de ML em tempo real.
- Selecionar e otimizar algoritmos de ML (como Redes Neurais Simples ou Classificadores Bayesianos) para operar dentro das restrições de memória e processamento de sistemas embarcados.
- Desenvolver um protótipo capaz de receber dados informativos dos equipamentos (ex: vibração ou temperatura) e realizar a inferência local para identificar padrões indicativos de falhas iminentes.
- Analisar a relação custo-benefício da solução proposta frente aos modelos tradicionais baseados em nuvem, focando na redução de custos operacionais.

1.4 Metodologia

A metodologia deste trabalho divide-se em etapas fundamentais que compreendem desde a concepção do modelo até sua execução física. Inicialmente, realiza-se a aquisição e o pré-processamento de uma base de dados pertinente ao problema, visando a extração de características relevantes. Em seguida, define-se a arquitetura de ML mais adequada às restrições do projeto, procedendo-se com o treinamento do modelo. Paralelamente, seleciona-se o microcontrolador com base nos requisitos computacionais necessários. Por fim, o modelo é implementado no hardware escolhido para a realização de testes em tempo real e a subsequente análise quantitativa dos resultados obtidos. A Figura 1 demonstra o fluxograma das atividades realizadas neste trabalho, sendo elas divida entre selecionar o modelo de ML, selecionar o MCU, encontrar um conjunto de dados e trata-lo, implementar o modelo de ML no MCU, testar e coletar os resultados.

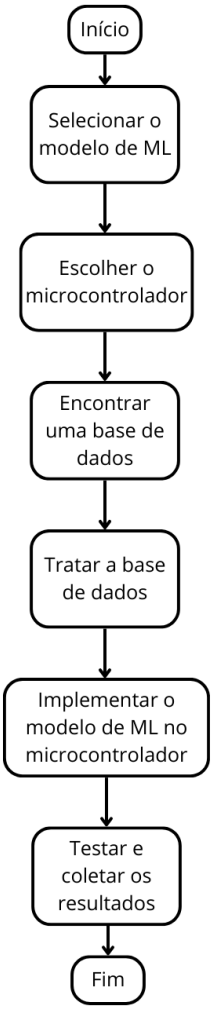


Figura 1 – Fluxograma das atividades.
Fonte: Do autor

1.5 Resultados e contribuições

Este trabalho contribui com a implementação de um protótipo de manutenção preditiva, por meio de uma arquitetura baseada no microcontrolador ESP32 e no classificador Naive Bayes. Os resultados quantitativos obtidos em 30 máquinas distintas demonstram a eficácia da solução, com a acurácia das inferências embarcadas entre 91% e 96%. A adoção de um limiar de decisão probabilístico de 80% permitiu otimizar o diagnóstico, resultando em uma precisão de até 96% e um *recall* de até 98%, o que comprova a alta sensibilidade do modelo na detecção de falhas reais com redução significativa de falsos positivos. Além disso, a estabilidade métrica foi confirmada por índices de F1-Score iguais ou superiores a 93%, evidenciando que a simplificação do modelo não comprometeu a robustez estatística necessária para o monitoramento industrial contínuo.

1.6 Organização do trabalho

A estrutura deste trabalho encontra-se organizada em cinco capítulos, descritos a seguir: O Capítulo 2 apresenta a fundamentação teórica necessária para o embasamento dos conceitos de ML e sistemas embarcados. O Capítulo 3 realiza uma revisão da literatura sobre trabalhos correlatos na área de manutenção preditiva. No Capítulo 4, é demonstrado a metodologia aplicada, descrevendo as etapas de desenvolvimento do protótipo, a seleção do hardware e a estratégia de otimização do modelo de ML. O Capítulo 5 é exibido e analisado os resultados obtidos nos testes experimentais, avaliando o desempenho da detecção local de falhas. Por fim, o Capítulo 6 consolida as considerações finais do estudo, discutindo o alcance dos objetivos propostos e sugerindo trabalhos futuros.

2 Referencial Teórico

Este capítulo aborda a Indústria 4.0 e como um de seus pilares, *Internet of Things (Internet das Coisas) (IoT)*, aliado ao uso de algoritmos de ML, define uma nova forma de realizar a manutenção e o controle de longevidade de equipamentos por meio da PdM na borda (edge). A Seção 2.1 define o conceito de Indústria 4.0 e contextualiza os sistemas IoT e MCU como elementos-chave para essa abordagem. A Seção 2.2 apresenta as definições e um comparativo entre a Manutenção Preditiva e a Reativa, explicando a implementação de algoritmos de ML em MCU. A Seção 2.3 explica o funcionamento de dois classificadores utilizados neste trabalho, o Naive Bayes e o Random Forest. A Seção 2.4 traz a explicação da técninca *Principal Component Analysis (Análise de Componentes Principais) (PCA)* usada para definir as variáveis usadas pelo modelo de ML escolhido. A Seção 2.5 demosntra o funcionamento da técnica de subamostragem utilizada para definir conjuntos de dados utilizados durante o treinamento. Por fim, a Seção 2.6 explica as métricas utilizadas na avaliação do desempenho do modelo de ML.

2.1 Industria 4.0

A Indústria 4.0 representa uma mudança de paradigma na manufatura global. Segundo Schwab (2016), esta fase não é apenas um prolongamento da terceira revolução industrial, mas sim uma mudança caracterizada pela velocidade e pelo impacto em toda a sociedade. Enquanto as revoluções anteriores focavam na mecanização e eletrônica, a Quarta Revolução foca na conectividade integral. Tecnicamente, o conceito baseia-se na implementação de CPS, onde dispositivos físicos e processos de software são integrados para monitorar e controlar o mundo físico de forma autônoma (KAGERMANN et al., 2013). O foco é a conectividade, permitindo que a produção deixe de ser uma sequência linear para se tornar uma rede dinâmica e interconectada. O principal objetivo da Indústria 4.0 não é somente a automação, mas a criação de sistemas de manufatura inteligentes e flexíveis.

Hermann, Pentek e Otto (2016) define que a essência desse sistema reside na capacidade de comunicação fluida entre todos os elementos da rede. O resultado final dessa integração é a chamada *Smart Factory* (Fábricas Inteligentes). Para Lasi et al. (2014), este ambiente é definido pela descentralização, onde os próprios sistemas tornam-se capazes de tomar decisões autônomas, otimizando a produção de forma independente e permitindo a customização em massa de produtos.

2.1.1 Sistemas IoT

Dentre os nove pilares da Indústria 4.0, os sistemas IoT atuam como o sistema nervoso da manufatura inteligente. Conceituada por Ashton (2009), a IoT define-se como uma rede de objetos físicos incorporados com sensores, software e outras tecnologias com o objetivo de conectar e trocar dados com outros dispositivos e sistemas pela internet. No contexto industrial, a IoT busca resolver o problema do

silêncio dos dados.

Porter e Heppelmann (2014) destaca que o "silêncio dos dados" refere-se à lacuna de visibilidade em processos onde a informação existe, mas não é capturada ou integrada ao sistema de tomada de decisão. Segundo Shannon (1948), a informação é a redução da incerteza. Portanto, o silêncio dos dados representa o estado máximo de incerteza operacional, onde eventos ocorrem de forma "invisível" aos gestores. Para Rüßmann et al. (2015), em modelos produtivos tradicionais, as informações sobre o estado de uma máquina ou a localização de um insumo eram isoladas ou dependiam de intervenção humana. A IoT automatiza essa coleta, permitindo o monitoramento em tempo real e a visibilidade total da cadeia produtiva.

O desafio central da implementação da IoT reside na ponte entre o mundo físico e o digital. Para que um objeto se torne "inteligente" e capaz de transmitir dados, ele necessita de uma unidade de processamento local que execute algoritmos de controle e protocolos de comunicação. Essa viabilidade técnica é sustentada pelo avanço dos MCU, dispositivos semicondutores compactos que funcionam como o "cérebro" embarcado de cada nó da rede IoT.

2.1.2 Microcontroladores

O MCU é definido por Dang (2022) como um sistema de processamento independente em um único circuito integrado, caracterizado pela integração monolítica de uma Unidade Central de Processamento (UCP), *Random Access Memory (Memória de Acesso Aleatório) (RAM)* para dados, *Read-Only Memory (Memória Somente de Leitura) (ROM/Flash)* para o programa e portas de entrada/saída. Para Dang (2022), diferentemente da computação de uso geral, esses dispositivos são componentes fundamentais de sistemas IoT, projetados para executar tarefas dedicadas e de tempo real sob restrições rígidas de energia e dimensões físicas.

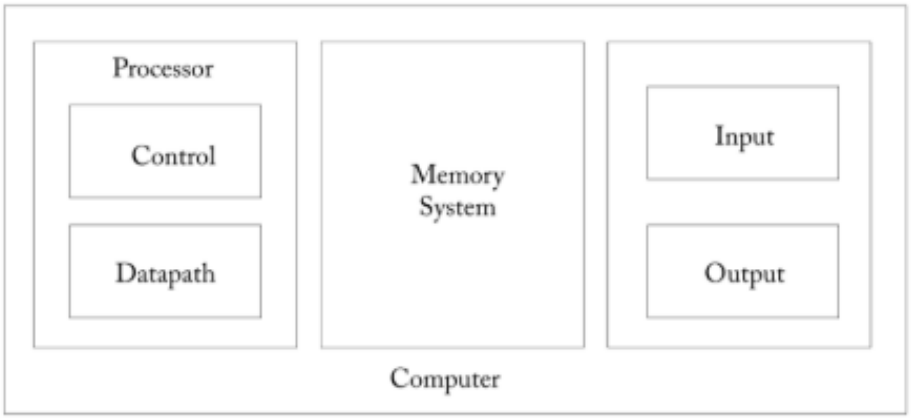


Figura 2 – Esquema básico de um MCU
Fonte: (DANG, 2022)

Para além das unidades básicas, MCU modernos incorporam uma vasta gama de periféricos de hardware que expandem sua capacidade de interação com o mundo físico. Entre esses, Marwedel (2021) destaca os conversores Analógico-Digitais, temporizadores de alta precisão, moduladores de largura de pulso e interfaces de comunicação serial. Essa integração permite que o MCU atue como o elo de tradução entre grandezas físicas e sinais digitais.

No contexto da Indústria 4.0, a evolução dos MCU viabilizou o paradigma de *Edge Computing*. Ao integrar unidades captadoras de informação e núcleos de processamento capazes de executar inferências de ML, MCU tornam-se dispositivos autônomos de tomada de decisão, responsáveis por quebrar o "silêncio dos dados" ao realizar a filtragem e análise preliminar da informação na origem, reduzindo a latência e a dependência de sistemas em nuvem e possibilitando a PdM.

2.2 Identificação de falhas em equipamentos de processos industriais

A integridade de ativos físicos em ambientes industriais é um dos pontos da eficiência operacional na Indústria 4.0. A identificação precoce de falhas não apenas evita paradas não planejadas (*downtime*), mas também mitiga riscos de acidentes de trabalho e danos ambientais (MOBLEY, 2002). Com a crescente complexidade dos sistemas de manufatura, métodos tradicionais de inspeção visual ou baseados apenas em tempo de uso tornaram-se insuficientes. O processo de identificação de falhas baseia-se na coleta contínua de variáveis críticas, tais como vibração, temperatura, pressão e corrente elétrica. A análise espectral de vibração, por exemplo, permite diagnosticar desalinhamentos e desgastes em rolamentos antes que a quebra ocorra (RANDALL, 2011).

Lee, Bagheri e Kao (2014) define que a simples detecção de uma falha atual evoluiu para a capacidade de prever quando ela ocorrerá. A PdM utiliza modelos estatísticos e

algoritmos de aprendizado de máquina para estimar a vida utíl restante de um componente. Este tema é fundamental para entender como o histórico de dados transforma a reação em antecipação. Tradicionalmente, os dados coletados são enviados para a nuvem para processamento. Entretanto, surge a necessidade de processamento local para respostas em milissegundos e economia de custos com infraestrutura em nuvem. É aqui que entra o TinyML. Esta tecnologia permite que algoritmos complexos de detecção de falhas sejam executados diretamente em MCU de baixo consumo, como ESP32 ou MCU da família Arduino, permitindo que o próprio controlador "decida" se há algo errado sem depender de conectividade constante (WARDEN; SITUNAYAKE, 2019).

2.2.1 Manutenção Reativa e Preditiva

A gestão da manutenção evoluiu de uma atividade meramente operacional para uma função estratégica centrada na confiabilidade, de acordo com Kardec e Nascif (2009). Para compreender a importância da identificação de falhas, é necessário distinguir as abordagens clássicas da manutenção industrial (MOBLEY, 2002).

A *Reactive Maintenance (Manutenção Reativa) (RM)*, frequentemente denominada "quebra-repara", é a estratégia onde a intervenção ocorre apenas após a falha funcional do equipamento. Segundo Mobley (2002), esta abordagem é a mais custosa a longo prazo devido às paradas de produção não planejadas, danos secundários a componentes adjacentes e custos elevados com horas extras e fretes de peças de reposição. Embora seja aceitável para equipamentos de baixa criticidade, sua aplicação em processos centrais compromete a competitividade.

Em contraste, a PdM baseia-se no monitoramento contínuo da condição real do ativo. De acordo com Randall (2011), a premissa é que a maioria das falhas não ocorre de forma instantânea, mas emite sinais de degradação que podem ser detectados precocemente. Ao analisar tendências de dados, a manutenção é agendada apenas quando estritamente necessário, maximizando a vida útil dos componentes e minimizando intervenções desnecessárias, típicas da manutenção preventiva baseada em tempo (LEE; BAGHERI; KAO, 2014). Abaixo, apresenta-se um comparativo técnico entre as duas abordagens, destacando o impacto na operação industrial:

Tabela 1 – Comparativo técnico entre as estratégias de manutenção industrial.

Característica	**RM**	**PdM**
Momento da Ação	Após a falha/quebra.	Antes da falha, baseada na condição.
Custo Operacional	Alto (paradas imprevistas).	Otimizado (intervenções planejadas).
Vida Útil do Ativo	Reduzida por danos colaterais.	Estendida pelo monitoramento constante.
Complexidade	Baixa (não exige sensores).	Alta (exige sensores e análise de dados).
Segurança	Menor (risco de falhas catastróficas).	Maior (ambiente controlado e monitorado).

Fonte: Elaborado pelo autor (2026), baseado em Mobley (MOBLEY, 2002) e Randall (RANDALL, 2011).

Como é possível observar, apesar de exigir maior controle sobre ambiente e investimento em equipamentos de monitoramento, a PdM apresenta muitas vantagens em relação a RM, principalmente o retorno do investimento em otimização e extensão da vida útil dos ativos.

2.2.2 Computação em Borda

Segundo Shi et al. (2016), a computação de borda, ou *edge computing*, é um paradigma arquitetônico de tecnologia da informação que aproxima o processamento, o armazenamento e a análise de dados do local exato onde eles são gerados. Historicamente, o modelo predominante de computação em nuvem exigia que todos os dados coletados por sistemas e dispositivos fossem transmitidos para data centers centralizados. Com o crescimento exponencial da geração de dados globais e o aumento da demanda por interações em tempo real, esse tráfego contínuo de informações pesadas começou a gerar gargalos críticos de latência, limitações na largura de banda das redes de comunicação e exigência um alto investimento em servidores centrais.

Para solucionar essas barreiras, a computação de borda implementa uma topologia de rede distribuída. Em vez de enviar o volume total de dados brutos para servidores distantes, a capacidade de processamento analítico é transferida para os roteadores ou para os próprios dispositivos físicos que compõem a extremidade da rede. Para Satyanarayanan (2017) essa mudança de paradigma reduz drasticamente a distância que a informação precisa percorrer, permitindo que algoritmos analisem o contexto local e executem ações de forma autônoma e quase instantânea, garantindo o funcionamento do sistema mesmo diante de instabilidades, ausência total de conexão com a internet externa e ausência de servidores centrais.

2.2.3 TinyML para Manutenção Preditiva

A implementação da PdM em larga escala enfrenta desafios significativos relacionados à infraestrutura de rede e ao custo de armazenamento de dados. Em um ambiente industrial típico, sensores de vibração de alta frequência podem gerar gigabytes de dados brutos diariamente. É neste gargalo que o TinyML surge como a tecnologia habilitadora para a próxima geração de sistemas de monitoramento.

O TinyML refere-se à intersecção de sistemas embarcados e aprendizado de máquina, permitindo que modelos de inferência sejam executados em hardware com recursos limitados, como MCU com poucos kilobytes de memória, se enquadrando na computação de borda (WARDEN; SITUNAYAKE, 2019). Ao integrar o TinyML a PdM, a análise de falhas deixa de ocorrer em servidores centrais e passa a ocorrer no próprio ativo. De acordo com conclusões tiradas da obra de Zhang, Yang e Wang (2020), a descentralização oferece três vantagens para a industria:

1. **Redução de Latência:** A detecção de uma falha iminente é instantânea, permitindo o desligamento automático de emergência sem depender da rede.
2. **Economia de Banda:** Apenas alertas ou metadados são enviados para o sistema supervisório, em vez do fluxo massivo de dados brutos.
3. **Privacidade e Segurança:** Os dados operacionais sensíveis permanecem dentro do dispositivo, reduzindo a superfície de ataque para invasões cibernéticas.

A viabilidade econômica da manutenção preditiva em todos os equipamentos de uma fábrica torna-se real com o uso de chips como o ESP32 ou da família Arduino. Estes componentes, que custam uma fração de um Controlador Lógico Programável (CLP) tradicional, são capazes de executar algoritmos para classificação de sinais (LEE; BAGHERI; KAO, 2014). Neste trabalho, será abordado o paradigma do *On-Device Training* que é uma subárea avançada do TinyML.

O TinyML abrange todo o ecossistema de aprendizado de máquina em hardware de baixo consumo, mas a literatura faz uma distinção clara entre as etapas de execução. Na arquitetura tradicional de TinyML, o modelo é treinado em ambientes de alto desempenho, como nuvem ou estações de trabalho, e posteriormente otimizado para execução em MCU. No entanto, o treinamento no próprio dispositivo (*on-device training*) é uma subárea avançada do TinyML que permite a personalização do modelo em tempo real (WARDEN; SITUNAYAKE, 2019).

Ao realizar o treinamento diretamente no chip, como um ESP32 ou família Arduino, o sistema de manutenção preditiva ganha a capacidade de aprender as especificidades de “ruído normal” de uma máquina específica, adaptando-se a variações que modelos

genéricos poderiam interpretar erroneamente como anomalias (ZHANG; YANG; WANG, 2020). Dessa forma, utilizar o MCU para treinar o modelo não apenas mantém o projeto dentro do escopo de TinyML, como também o coloca na fronteira tecnológica da Indústria 4.0, ao permitir que o ativo industrial possua um aprendizado contínuo e personalizado (LEE; BAGHERI; KAO, 2014).

2.3 Tarefa de classificação

A classificação é uma subárea de ML supervisionado que visa atribuir uma categoria discreta a uma entrada de dados. Conforme definido por Mitchell (1997), o problema de aprendizagem de uma tarefa de classificação consiste em aproximar uma função de alvo $f : X \to Y$, onde X é o espaço de instâncias (características) e Y é um conjunto finito de rótulos simbólicos. Segundo Hastie, Tibshirani e Friedman (2009), o objetivo principal é construir uma regra de decisão que divida o espaço de entrada em regiões identificadas com cada classe. No caso de uma classificação binária, o modelo busca uma fronteira de decisão que separe as observações onde $Y = 0$ daquelas onde $Y = 1$.

2.3.1 Naive Bayes

O classificador Naive Bayes é um algoritmo de ML supervisionado baseado na aplicação do Teorema de Bayes, sob a suposição de independência condicional entre as variáveis. Segundo Mitchell (1997), essa "ingenuidade", daí o termo *Naive*, simplifica drasticamente o cálculo das probabilidades posteriores, tornando o modelo computacionalmente eficiente e eficaz mesmo em grandes dimensões. Seu funcionamento se resume a: Dado um vetor de características $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, o objetivo é encontrar a probabilidade de que este vetor pertença a uma classe específica C_k. Pelo Teorema de Bayes, temos:

$$P(C_k|\mathbf{x}) = \frac{P(C_k)P(\mathbf{x}|C_k)}{P(\mathbf{x})} \tag{2.1}$$

Onde:

- $P(C_k|\mathbf{x})$ é a probabilidade posterior;
- $P(C_k)$ é a probabilidade *a priori* da classe;
- $P(\mathbf{x}|C_k)$ é a verossimilhança (*likelihood*);
- $P(\mathbf{x})$ é a evidência, que atua como uma constante de normalização.

Assumindo a independência condicional entre as características x_i conforme proposto por Hastie, Tibshirani e Friedman (2009), a verossimilhança pode ser decomposta no produto das probabilidades individuais:

$$P(\mathbf{x}|C_k) = \prod_{i=1}^{n} P(x_i|C_k) \tag{2.2}$$

2.3.2 Naive Bayes Gaussiano

Quando lidamos com dados contínuos, a abordagem comum é assumir que os valores associados a cada classe seguem uma distribuição normal ou Gaussiana (BISHOP, 2006). A probabilidade condicional é então calculada pela função de densidade de probabilidade da distribuição normal:

$$P(x_i|C_k) = \frac{1}{\sqrt{2\pi\sigma_{ki}^2}} \exp\left(-\frac{(x_i - \mu_{ki})^2}{2\sigma_{ki}^2}\right) \tag{2.3}$$

Nesta equação, os parâmetros são estimados a partir dos dados de treinamento:

- μ_{ki}: representa a média da característica i para a classe C_k;
- σ_{ki}^2: representa a variância da característica i para a classe C_k.

O classificador final seleciona a classe que maximiza a probabilidade posterior. Para evitar problemas de precisão numérica (*underflow*), comum em processadores de 32 bits ao manipular multiplicações de probabilidades extremamente pequenas, é comum utilizar o logaritmo da probabilidade, transformando o produto em uma soma:

$$\hat{y} = \ln P(C) + \sum_{i=1}^{n} \ln P(x_i|C) \tag{2.4}$$

Esta abordagem permite que o modelo seja robusto e escalável para aplicações práticas, como detecção de falhas e classificação de sinais industriais.

2.3.3 Random Forest

O algoritmo Random Forest, proposto originalmente por Breiman (2001), é um método de aprendizado de máquina por conjunto que opera construindo uma multiplicidade de árvores de decisão durante o treinamento. A predição final é obtida por votação majoritária (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

O Random Forest é definido como um preditor que consiste em uma coleção de classificadores estruturados em árvores $\{T(x, \Theta_b), b = 1, \ldots, B\}$, onde os $\{\Theta_b\}$ são vetores aleatórios independentes e identicamente distribuídos (BREIMAN, 2001). Dado um conjunto de treinamento $Z = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, o algoritmo gera B amostras de *bootstrap* Z^*. Para cada amostra, uma árvore T_b é treinada. A votação é obtida através de:

$$\hat{C}_{rf}^{B}(x) = \sum_{b=1}^{B} I(T_b(x, \Theta_b) = k) \tag{2.5}$$

Onde:

- $\hat{C}_{rf}^{B}(x)$: Classe predita pelo modelo final para a entrada x.
- B: Número total de árvores na floresta.
- $T_b(x, \Theta_b)$: Predição da b-ésima árvore individual.
- $I(\cdot)$: Função indicadora (retorna 1 se a condição for verdadeira, 0 caso contrário).
- k: Índice representando a classe (ex: 0 para Normal, 1 para Falha).

O diferencial do Random Forest é a restrição do espaço de busca no momento da divisão de cada nó. Em vez de considerar todos os p atributos, o algoritmo seleciona aleatoriamente $m \approx \sqrt{p}$ variáveis. O objetivo é minimizar a impureza do nó, frequentemente utilizando o Índice de Gini (G):

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{2.6}$$

Onde:

- G: Índice de Gini (medida de impureza do nó).
- K: Número total de classes no problema.
- m: Identificador do nó atual na árvore de decisão.
- $\hat{p}_{mk}$: Proporção de instâncias da classe k presentes no nó m.

A divisão ótima é aquela que maximiza a redução da impureza total após divisão do nó (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

2.4 Análise de Componentes Principais (PCA)

A PCA é uma técnica estatística multivariada fundamental para a redução de dimensionalidade e visualização de dados. O objetivo central da PCA é transformar um conjunto de variáveis possivelmente correlacionadas em um conjunto de valores de variáveis linearmente não correlacionadas denominadas componentes principais (JOLLIFFE, 2002).

Segundo Hotelling (1933) a técnica baseia-se na decomposição espectral da matriz de covariância. Dado um conjunto de dados X, buscamos encontrar as direções de máxima variância. Matematicamente, isso envolve resolver o problema de autovalores e autovetores:

$$C\mathbf{v} = \lambda\mathbf{v} \tag{2.7}$$

Onde:

- C é a matriz de covariância dos dados.
- λ representa os autovalores (variância explicada).
- $\mathbf{v}$ representa os autovetores (componentes principais).

A interpretação dos resultados da PCA é auxiliada pelo Gráfico de Cargas. O Gráfico de Cargas é a representação visual dos pesos que cada variável original possui na construção dos componentes principais. Matematicamente, a carga da j-ésima variável no i-ésimo componente principal indica a correlação entre essa variável e o componente. Quanto mais distante um ponto (variável) estiver da origem (0,0), maior é a sua importância para a definição daquele plano de componentes.

2.5 Random Undersampling

Para He e Garcia (2009) *Random Undersampling* (Subamostragem Aleatória) é uma técnica de reamostragem a nível de dados que visa equilibrar a distribuição de classes. O método consiste em remover aleatoriamente exemplos da classe majoritária até que uma proporção desejada entre as classes seja alcançada. Matematicamente, se temos um conjunto de dados S composto por um subconjunto majoritário S_{maj} e um minoritário S_{min}, onde $|S_{maj}| \gg |S_{min}|$, a técnica busca criar um novo conjunto $S'_{maj} \subset S_{maj}$ tal que:

$$|S'_{maj}| = |S_{min}| \times \alpha \tag{2.8}$$

Onde α é um hiperparâmetro que define a razão final entre as classes, frequentemente definido como 1 para 1 equilíbrio perfeito. De acordo com Chawla (2005), as principais características da técnica são:

- **Eficiência Computacional:** Ao reduzir o tamanho do conjunto de dados total, o tempo de treinamento do modelo é significativamente diminuído, o que é vantajoso em sistemas com grandes volumes de dados.
- **Simplicidade:** Por ser um método não-heurístico, sua implementação é direta e não requer pressupostos complexos sobre a distribuição dos dados.

- **Risco de Perda de Informação:** A principal desvantagem, conforme apontado por He e Garcia (2009), é que a remoção aleatória pode descartar exemplos potencialmente importantes que ajudariam o classificador a definir o limite de decisão da classe majoritária.

2.6 Métricas

Para avaliar o desempenho do modelo nas inferencias embarcadas foram utilizadas quatro métricas: Precisão, Recall, F1-Score e Acurácia. Onde:

VP **(Verdadeiro Positivo):** Quantidade de instâncias de falha corretamente identificadas pelo modelo.

VN **(Verdadeiro Negativo):** Quantidade de instâncias de operação normal corretamente identificadas.

FP **(Falso Positivo):** Ocorrências onde o modelo indicou uma falha inexistente (Alarme Falso).

FN **(Falso Negativo):** Ocorrências onde uma falha real não foi detectada pelo modelo (Risco Crítico).

A acurácia representa a fração de previsões corretas entre todas as previsões feitas.

$$\text{Acurácia} = \frac{VP + VN}{VP + VN + FP + FN} \tag{2.9}$$

A precisão indica a proporção de instâncias positivas reais entre todas as instâncias classificadas como positivas pelo modelo.

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

O *Recall* (ou revocação) mede a capacidade do modelo de identificar todas as instâncias da classe de interesse (falhas). Para o contexto de manutenção preditiva, esta métrica é crítica, pois um baixo *recall* significa que falhas estão ocorrendo sem serem detectadas.

$$\text{Recall} = \frac{VP}{VP + FN} \tag{2.11}$$

O F1-Score é a média harmônica entre a precisão e o *recall*. Segundo Powers (2011), essa métrica é mais robusta que a acurácia para avaliar o equilíbrio entre falsos positivos e falsos negativos.

$$\text{F1-Score} = 2 \times \frac{\text{Precisão} \times \text{Recall}}{\text{Precisão} + \text{Recall}} \tag{2.12}$$

2.7 Considerações finais

Este capítulo estabeleceu a fundamentação teórica necessária para a compreensão do ecossistema onde o trabalho está inserido, exibindo os conceitos de industria 4.0, sistemas IoT, MCU, manutenção preditiva, TinyML, classificador e funcionamento do Naive Bayes, PCA, Random Forest e métricas que guiarão o desenvolvimento das tarefas subsequentes.

3 Trabalhos relacionados

A seguir, são discutidas pesquisas que serviram de base teórica ou que apresentam propostas similares à desenvolvida nesta monografia, comparando suas metodologias e resultados. O objetivo desta análise é identificar as abordagens mais utilizadas para solucionar o problema de detecção de falhas em equipamentos industriais e destacar as lacunas que motivaram o desenvolvimento deste trabalho.

O trabalho desenvolvido por Seidel et al. (2024) foca na detecção de falhas mecânicas (desbalanceamento e desalinhamento) em motores elétricos de indução trifásicos utilizando modelos de ML embarcados em MCU. A motivação surge no contexto da Indústria 4.0 e da necessidade de PdM para evitar paradas e prejuízos. O estudo utiliza dados de vibração coletados por um sensor IoT comercial (WEGscan100) em um motor trifásico real.

A principal contribuição é a investigação do uso de modelos de ML (Árvore de Decisão, SVM e One-Class SVM) executados diretamente no MCU do sensor, explorando as vantagens do processamento na borda, como menor latência e independência de rede, apesar das restrições de memória, processamento e energia. Os modelos supervisionados (Árvore de Decisão e SVM) apresentaram acurácia superior a 90% nos dados de treinamento e entre 71.5% e 83.3% na inferência embarcada. O modelo não supervisionado (One-Class SVM) mostrou alta dependência dos parâmetros, com boa acurácia nos dados de teste (98.2%) mas baixa nos dados normais durante a inferência (9.5%).

O trabalho de Seidel et al. (2024) converge com a proposta desta pesquisa ao alinhar-se com as motivações da Indústria 4.0, validando a importância do processamento na borda (*Edge Computing*) para garantir baixa latência e independência de rede em ambientes restritos. No entanto, os resultados apresentados evidenciam uma lacuna significativa que é a eficácia real na inferência embarcada (71,5% - 83,3%), além da instabilidade observada em modelos não supervisionados. Diante desse cenário, o presente trabalho busca avançar o estado da arte propondo o uso de classificadores computacionalmente mais “baratos” (Classificadores como o Nave Bayes), visando não apenas a redução do custo de processamento, mas, primordialmente, o aumento da robustez e da acurácia durante a fase crítica de inferência no dispositivo. Outra lacuna que este trabalho busca preencher é a utilização de mais de uma variável de análise, levando em conta informações de temperatura, consumo de óleo, tempo de operação e consumo de ernegia.

Por sua vez, o trabalho realizado por Franco (2020) propõe um sistema de PdM utilizando ML em um sistema embarcado. O objetivo central é avaliar a viabilidade de executar algoritmos de predição em um dispositivo embarcado (Raspberry Pi 3) sem a necessidade de um computador, comparando o desempenho com a execução em um

computador. Para isso, utiliza um dataset público da Kaggle sobre falhas no sistema de pressão de ar em caminhões Scania e testa cinco algoritmos (Rede Neural Densa, SVM, Random Forest, KNN e Regressão Logística). A metodologia detalha os cenários de teste, o tratamento de dados (normalização, exclusão por correlação) e a avaliação baseada em tempo de processamento e precisão.

Os resultados mostram que, embora o tempo de processamento seja significativamente maior no sistema embarcado, a precisão dos algoritmos não é impactada, concluindo pela viabilidade da proposta. Este trabalho traz uma análise importante que demonstra que os sistemas IoT podem entregar bons resultados. Nesse sentido, a transição de algoritmos de ML para plataformas de hardware limitado representa um avanço estratégico na democratização da PdM. A capacidade de operar com eficiência similar à de um computador convencional, porém em um ecossistema de baixo custo, permite que pequenas e médias indústrias implementem monitoramento de ativos que antes eram restritos a grandes corporações com alto poder de investimento.

A obra de Franco (2020) se relaciona com o desenvolvimento deste trabalho ao demonstrar que mesmo algoritmos de alta robustez são capazes de manter elevados níveis de desempenho em sistemas embarcados. Contudo, ao analisar o hardware utilizado pelo referido autor, observa-se que o custo do MCU — aproximadamente R$ 400,00 no mercado nacional (Raspberry Pi Foundation, 2026) — torna-se elevado. Em contraste, este trabalho propõe a utilização de dispositivos com melhor relação custo-benefício, como a linha ESP32, visando reduzir o custo da montagem do dispositivo IoT.

Por fim, o estudo conduzido por Graff (2019) apresenta um sistema de *Industrial Internet of Things (Internet das Coisas Industrial) (IIoT)* voltado para a predição de anomalias por meio de técnicas de Aprendizado de Máquina. Alinhado aos preceitos da Indústria 4.0, o objetivo central é viabilizar o monitoramento inteligente em pequenas e médias empresas que possuem restrições orçamentárias. A metodologia proposta utiliza uma arquitetura híbrida, empregando computação de borda para a coleta e pré-processamento de dados e a plataforma *Azure Machine Learning Studio* para o treinamento e execução dos modelos de ML. O estudo de caso foca no monitoramento de grandezas elétricas (tensão e corrente) em um ambiente laboratorial, utilizando o algoritmo KNN para a detecção de padrões.

Os resultados obtidos demonstram a eficácia da técnica de clusterização, sendo capaz de classificar com precisão o estado de operação do sistema em quatro níveis: desligado, normal, atenção e anormal. Além disso, a integração com uma interface de usuário desenvolvida em *Node-RED* permitiu o acompanhamento das métricas e do status operacional em tempo real, validando a funcionalidade do ecossistema proposto.

A pesquisa de Graff (2019) corrobora a viabilidade de aplicar modelos de *Machine Learning* para a PdM de baixo custo no cenário industrial. Entretanto, o MCU utilizado

foi um Raspberry Pi - aproximadamente R$ 400,00 no mercado nacional (Raspberry Pi Foundation, 2026) -, cujo valor de mercado pode ser um limitador para a escala massiva de nós sensores. Nesse ponto, o presente trabalho diferencia-se e avança ao propor a utilização da linha ESP32. O uso deste MCU não apenas reduz significativamente o custo de hardware em comparação ao Raspberry Pi, como também oferece o poder de processamento necessário para as tarefas de predição local, otimizando a relação custo-benefício para a implementação de sistemas de predição de falhas em larga escala.

4 Desenvolvimento e Metodologia

Este capítulo aborda todo o processo de desenvolcimento do trabalho e as metodologias aplicadas. A Seção 4.1 apresenta os crivos que foram usados na seleção do MCU e um comparativo entre o MCU escolhido e seus concorrentes de mesmo preço do mercado. A Seção 4.2 demonstra qual foi os critérios de escolha do modelo de ML e um comparativo entre o modelo escolhido e outros modelos com mesmo fim. Por sua vez, a Seção 4.3 destrincha o processo de busca por um *Dataset* que atenda as necessidades do trabalho. A Seção 4.4 realiza a análise exploratória do conjunto encontrado, descrevendo o que representa cada dimensão e sua distribuição, além de descrever o processo de preparação dos dados para as inferências embarcadas. A Seção 4.5 descreve o processo de seleção de dimensões para as inferências embarcadas. A Seção 4.6 mostra a técninca de balanceamento utilizada na base de dados, a construção de subamostras para as inferências e o *setup* de treinamento e validação do modelo. Por fim, a Seção 4.7 mostra pseudocódigos para descrever o algoritmo implementado.

4.1 Escolha do microcontrolador ESP32

A seleção do ESP32 como MCU é baseada na combinação entre viabilidade técnica e econômica. O componente destaca-se pelo baixo custo e pela compatibilidade nativa com ambientes de desenvolvimento voltados para ML, como MicroPython e TensorFlow Lite. Suas especificações de hardware, incluindo 512KB de memória RAM, 4MB de ROM/Flash e alta velocidade de processamento, oferecem a infraestrutura necessária para alocar tanto o dataset quanto o modelo preditivo (Espressif Systems, 2024).

A escolha foi reforçada pela comparação entre outros MCU concorrentes ao ESP32. Abaixo, segue uma tabela com dados de comparação técnica entre o MCU escolhido e outros na mesma faixa de preço.

Tabela 2 – Comparativo entre ESP32 e MCU concorrentes do mercado

Característica	ESP32	RP2040	STM32F401	Arduino Nano
Processamento	240 MHz (Dual-Core)	133 MHz (Dual-Core)	84 MHz (Single-Core)	20 MHz (8-bit)
Arquitetura	Xtensa LX6 (32-bit)	ARM Cortex-M0+	ARM Cortex-M4	ATMega4809
Memória RAM	520 KB	264 KB	64 KB	6 KB
Memória ROM/Flash	4 MB	2 MB	256 KB	48 KB
Preço Médio	R$ 30,00 - R$ 40,00	R$ 35,00 - R$ 45,00	R$ 35,00 - R$ 50,00	R$ 45,00 - R$ 150,00*

Fonte: Do autor, baseado em: (ARDUINO SA, 2024), (RASPBERRY PI LTD, 2024), (STMICROELECTRONICS, 2025) e (Espressif Systems, 2024)

Ao analisar a tabela, fica evidente que o ESP32 oferece:

1. **Paralelismo e frequência de inferência:** Com o clock de 240 MHz e arquitetura Dual-Core, o ESP32 permite o isolamento de tarefas. Enquanto um núcleo gerencia a coleta de dados, o segundo núcleo pode ser dedicado a execução de inferência. Isso reduz drasticamente a latência de predição.

2. **Memória processamento de inferências:** A inferência de modelos demanda espaço para armazenar os dados. Os 520 KB de RAM do ESP32 permitem processar o algoritmo de ML com maior facilidade.

3. **Armazenamento de datasets maiores:** O armazenamento de 4 MB ROM/Flash facilita o armazenamento de maior quantidade de dados. Isso é fundamental para técnicas de classificação realizadas na borda, onde o espaço de 48 KB de um Arduino Nano, por exemplo, seria um limitador.

4. **Otimização via instruções de hardware:** A arquitetura Xtensa LX6 do ESP32 possui suporte a instruções que aceleram operações matemáticas comuns em ML, como multiplicações, necessárias para algoritmos de classificação.

A Figura 3 mostra uma imagem do MCU ESP32.

Figura 3 – Microcontrolador ESP32
Fonte: Imagem da internet

4.2 Escolha do modelo de Machine Learning

A definição do modelo de ML se deu por base em duas principais exigências: baixo custo computacional, já que os MCU não possuem o poder de processamento de um computador convencional, e desempenho com poucos dados, pois o MCU possui capacidade de armazenamento restrita.

Dessa forma, o modelo escolhido foi o classificador *Naive Bayes*. A escolha é fundamentada pela obra de Hand e Yu (2001) que define o algoritmo como o equilíbrio entre simplicidade arquitetural e robustez preditiva, apresentando uma solução onde a eficiência computacional é prioritária. Diferentemente de outros modelos clássicos que demandam otimizações complexas ou custosos cálculos de distância, como SVM e KNN, o *Naive Bayes* oferece um custo computacional linear, garantindo agilidade tanto na etapa de treinamento quanto na classificação (ZHANG, 2004).

Essas características, aliadas a uma implementação descomplicada e baixa sensibilidade à "maldição da dimensionalidade", um fenômeno que exige grandes quantidades de dados para o treinamento, permitindo que o modelo alcance bons resultados consumindo uma fração dos recursos de processamento exigidos por algoritmos de maior complexidade estrutural (MITCHELL, 1997). Abaixo, segue uma tabela comparativa entre o *Naive Bayes* e outros classificadores.

Tabela 3 – Comparativo entre classificadores: Naive Bayes, KNN e SVM

Critério	**Naive Bayes (Escolhido)**	**KNN**	**SVM**
Custo Computacional	**Baixo** (Linear). Realiza contagens simples de probabilidade.	**Alto** na classificação. Exige cálculo de distância para todos os pontos.	**Alto** no treinamento. Exige resolução de problemas de otimização quadrática.
Armazenamento	**Baixo**. Armazena apenas estatísticas (médias/probabilidades).	**Alto**. Precisa armazenar todo o dataset na memória para funcionar.	**Moderado**. Armazena apenas os vetores de suporte.
Sensibilidade à Dimensionalidade	**Robusto**. Trata variáveis de forma independente.	**Muito Sensível**. Sofre severamente com a maldição da dimensionalidade.	**Moderado**.Pode ser complexo ajustar.
Complexidade de Implementação	**Simples**. Matemática direta e fácil de codificar em baixo nível.	**Simples**, porém custoso computacionalmente.	**Complexa**. Matemática avançada difícil de implementar do zero.

Fonte: Do autor, baseado em: (HAND; YU, 2001), (BISHOP, 2006), (CORTES; VAPNIK, 1995) e (COVER; HART, 1967)

A seleção do algoritmo *Naive Bayes* como o classificador central deste projeto não se limita apenas à sua simplicidade, mas à sua eficiência operacional em ambientes de computação restrita. Sendo destacados os seguintes pontos:

1. **Independência de atributos e escalabilidade:** A premissa de independência condicional do Naive Bayes (BISHOP, 2006) permite que o modelo trate cada variável do dataset de forma isolada. Isso simplifica o cálculo da verossimilhança, exigindo apenas operações aritméticas básicas que são processadas com extrema rapidez pelo núcleo Xtensa do ESP32.

2. **Baixo consumo de memória:** Diferente do KNN (COVER; HART, 1967), que requer alocação dinâmica e volumosa para o armazenamento de vizinhos, o Naive Bayes opera com um modelo probabilístico compacto. Uma vez treinado, o classificador resume-se a uma tabela de pesos e probabilidades que pode ser armazenada na memória ROM/Flash do dispositivo, preservando os 520 KB de RAM.

3. **Baixo custo computacional:** Em aplicações de monitoramento ou predição contínua, a latência de inferência é crítica. O Naive Bayes tem baixo custo computacional, permitindo que o ESP32 realize predições rapidamente, superando a complexidade de busca do KNN ou a carga matemática de vetores de suporte do SVM (CORTES; VAPNIK, 1995).

Em suma, o Naive Bayes atende ao requisito de viabilidade econômica e técnica, transformando o ESP32 em um componente capaz de realizar predições locais robustas sem a dependência de processamento em nuvem.

4.3 Conjunto de dados

O primeiro passo foi buscar a criação de um conjunto de dados que possuísse as informações necessárias para o treinamento de um algoritmo de ML. Contudo, essa possibilidade foi descartada pois seria custoso e demandaria muito tempo. Para a criação desse conjunto de dados, seria preciso obter informações de sensores de diversos tipos, como sensores de vibração, temperatura, nível de olho, tensão, som, extraidos de diversos tipos de máquinas e equipamentos industriais. Seria preciso ter parceria com uma ou mais industrias para coleta de dados e aquisição de todo o material necessário para isso.

Dessa forma, a solução encontrada foi a procura por um conjunto de dados já existente que cumprisse alguns requisitos: Possuir dados de diversos tipos de máquinas e equipamentos para tornar a solução utilizável em vários equipamentos industriais, possuir informações de coleta de diversos tipos de sensores, ser suficientemente grande para suprir a necessidade de dados para teste, treinamento e validação do modelo. A busca se deu em plataformas de disponibilização de dados gratuitas como *Kaggle* (Kaggle, 2025) e *UCI Machine Learning Repository* (UCI Machine Learning Repository, 2025).

4.3.1 Conjunto de dados escolhido

Após realizar diversas buscas nos sites citados, foi encontrado um *Dataset* que atende todos os requisitos pré-estabelecidos. A base em questão é a *Industrial IoT Dataset (Synthetic)* (Can Özensoy, 2025) que traz informações sintéticas de coleta de sensores de diversas máquinas e equipamentso industriais com 22 dimensões de dados.

A base foi desenvolvida para aplicações de ML na Indústria, focando especificamente em manutenção preditiva e detecção de anomalias. Composta por registros de 500.000 máquinas simuladas em um ambiente de fábrica, abrangendo mais de 30 tipos de equipamentos realistas e fornece dados essenciais de sensores, como temperatura, vibração, som, energia, níveis de fluidos, e métricas operacionais específicas, como pressão hidráulica. A base também oferece uma legenda que classifica os dados como dados de máquinas que vão falhar em até 7 dias e dados de máquinas em plena operação.

A utilização de um conjunto de dados sintético nesta pesquisa é justificado pela escassez e pelas barreiras de acesso a dados reais em fontes da literatura, repositórios públicos e dados de coletas reais. Conforme dito por Jordan e Mitchell (2015), a dificuldade em obter grandes volumes de dados de alta qualidade é um dos principais gargalos para o desenvolvimento de modelos robustos. Segundo Goodfellow, Bengio e Courville (2016) muitos conjuntos de dados brutos são protegidos por leis de privacidade e sigilo industrial, o que impede sua disponibilização integral. Além disso, a literatura muitas vezes descreve metodologias, mas não fornece os dados brutos para replicação direta.

Outro fator determinante é o custo e a complexidade experimental. Em cenários de

manutenção preditiva, a obtenção de dados de falhas reais exige que equipamentos operem até o estado crítico, resultando em altos custos e riscos operacionais. Logo, a simulação e a geração de dados sintéticos tornam-se ferramentas essenciais para validar algoritmos de detecção de falhas.

Abaixo, a tabela 4 apresenta um dicionário de dados do conjunto de dados escolhido, contendo nome da variável, tipo de variável, em qual escala os dados estão e uma breve descrição do que as variáveis representam.

Dicionário de Dados

Tabela 4 – Dicionário de dados

Nome	**Tipo**	**Escala**	**Descrição**
Machine_ID	String	ID	Identificador único do equipamento.
Machine_Type	String	Texto	Tipo/Categoria da máquina industrial.
Installation_Year	Int	Ano (AAAA)	Ano em que a máquina foi instalada.
Operational_Hours	Float	Horas	Total de horas de operação acumuladas.
Temperature_C	Float	°C (Celsius)	Temperatura operacional do equipamento.
Vibration_mms	Float	mm/s	Nível de vibração (velocidade eficaz).
Sound_dB	Float	dB (Decibéis)	Intensidade do ruído emitido.
Oil_Level_pct	Float	% (0-100)	Nível de óleo lubrificante disponível.
Coolant_Level_pct	Float	% (0-100)	Nível de fluido de arrefecimento.
Power_Consumption_kW	Float	kW	Consumo de potência ativa.
Failure_History_Count	Int	Contagem	Total de falhas registradas anteriormente.
AI_Supervision	Boolean	T/F	Se a máquina é monitorada por IA.
Error_Codes_Last_30_Days	Int	Unidade	Quantidade de erros reportados no mês.
Remaining_Useful_Life_days	Float	Dias	Vida Útil Remanescente estimada (RUL).
Failure_Within_7_Days	Boolean	T/F	Indicador se a máquina falhou na semana seguinte.
Laser_Intensity	Float	W/cm²	Intensidade do laser (se aplicável).
Hydraulic_Pressure_bar	Float	bar	Pressão do sistema hidráulico.
Coolant_Flow_L_min	Float	L/min	Vazão do líquido de arrefecimento.
Heat_Index	Float	Índice	Cálculo de estresse térmico do componente.
AI_Override_Events	Int	Contagem	Vezes que o operador interviu na IA.

Fonte: Do autor

Baseado no dicionário a cima, o conjunto de dados utilizado neste projeto é extremamente completo, destacando:

- **Identificação:** A variável `Machine_Type` permitem a segmentação dos ativos por categoria (como *Mixers* ou *Chillers*).
- **Monitoramento Sensorial:** O *dataset* possui informações sobre grandezas físicas:
  - **Térmica:** `Temperature_C` monitora o estresse térmico do motor e componentes internos.
  - **Mecânicas:** `Vibration_mms` (mm/s) e `Sound_dB` (dB) detectam anomalias como desalinhamentos ou falhas de rolamento.
  - **Fluídicas:** `Hydraulic_Pressure_bar` (bar) e `Coolant_Flow_L_min` (L/min) garantem que os sistemas de suporte e arrefecimento operem dentro dos limites nominais.
- **Eficiência e Consumo:** A variável `Power_Consumption_kW` (kW) monitora o consumo de potência ativa. Padrões anômalos de consumo são indicadores frequentes de sobrecarga.

4.4 Análise exploratória e tratamento do conjunto de dados

O tratamento de dados (ou pré-processamento de dados) é considerado a etapa mais crítica em qualquer projeto ML. A importância reside no princípio fundamental de *"Garbage In"*, *"Garbage Out"* (Lixo entra, lixo sai): se os dados de entrada forem de má qualidade, os resultados da análise ou as previsões do modelo serão inevitavelmente falhos, independentemente da sofisticação dos algoritmos utilizados. Logo, essa etapa se torna parte fundamental e indispensável do trabalho. Para realizar essa etapa tão importante, é necessário realizar uma análise exploratória.

Toda a etapa foi realizada utilizando a linguagem de programação *Python*, as bibliotecas de ciência de dados *Pandas*, *Numpy*, *sklearn*, *matplotlib*, *seaborn*, e a plataforma de desenvolvimento *Google Colab*.

4.4.1 Tabela de sumarização inicial do conjunto de dados

A Tabela 5 apresenta o sumário estatístico inicial do conjunto de dados utilizado neste trabalho. Esta etapa de sumarização é fundamental para avaliar a necessidade de realizar tratamentos de valores nulos, transformações em dados de tipo não numérico e para avaliar a necessidade de colocar todos os dados na mesma escala.

Tabela de Sumarização

Tabela 5 – Sumarização Inicial dos dados

Variável	**Tipo**	**Nulos (%)**	**Média**	**Mediana**	**Desvio Padrão**
AI_Override_Events	int64	0.00	0.60	0.00	1.20
AI_Supervision	bool	0.00	0.30	0.00	0.46
Coolant_Flow_L_min	float64	90.88	39.98	40.03	9.99
Coolant_Level_pct	float64	0.00	64.11	65.01	23.15
Error_Codes_Last_30_Days	int64	0.00	3.00	3.00	1.73
Failure_History_Count	int64	0.00	2.00	2.00	1.41
Failure_Within_7_Days	bool	0.00	0.06	0.00	0.24
Heat_Index	float64	90.96	499.76	499.77	50.01
Hydraulic_Pressure_bar	float64	93.93	119.92	119.86	15.02
Installation_Year	int64	0.00	2019.96	2020.00	11.83
Laser_Intensity	float64	96.97	75.00	74.95	10.04
Last_Maintenance_Days_Ago	int64	0.00	182.26	182.00	105.55
Machine_ID	object	0.00	—	—	—
Machine_Type	object	0.00	—	—	—
Maintenance_History_Count	int64	0.00	5.00	5.00	2.24
Oil_Level_pct	float64	0.00	69.46	70.03	18.85
Operational_Hours	int64	0.00	50012.42	49973.00	28861.85
Power_Consumption_kW	float64	0.00	149.92	149.95	79.96
Remaining_Useful_Life_days	float64	0.00	452.42	451.00	288.97
Sound_dB	float64	0.00	75.00	75.00	9.99
Temperature_C	float64	0.00	60.00	60.00	14.99
Vibration_mms	float64	0.00	9.99	10.00	5.00

Fonte: Do autor

4.4.2 Tratamento de valores nulos e de tipo não numérico

Ao analisar a tabela 5, fica evidente que o conjunto de dados necessitam de pré-processamento. Primeiramente, destaca-se a alta quantidade de valores nulos em variáveis como Coolant_Flow_L_min (90,88%), Heat_Index (90,96%), Hydraulic_Pressure_bar (93,93%) e Laser_Intensity (96,97%). Essa característica sugere que essas variáveis vem de tipos de máquinas específicos, não estando disponíveis em toda as máquina do conjunto. Exemplo: Não faz sentido um forno industrial possuir a medição da pressão da prensa hidráulica (Hydraulic_Pressure_bar) pois o forno não possui essa função. Dessa forma,

todos os valores faltantes foram preenchidos com o valor zero.

Além da questão de valores faltantes, a tabela revela a necessidade de transformações de tipo para viabilizar o treinamento do algoritmo de ML. Variáveis identificadas como *booleanos* (AI_Supervision, Failure_Within_7_Days) e *object* (Machine_ID, Machine_Type) não podem ser processadas diretamente pelo modelo de classificação escolhido por ele ser matemático, exigindo entradas numéricas. Portanto, será essencial a aplicação de técnicas de codificação, como *One-Hot Encoding* para as categorias nominais e conversão binária (0 ou 1) para os *booleanos*, garantindo que toda a informação possa ser usado pelo modelo.

Para tratar os valores *booleanos* foi necessário apenas fazer um *Cast* nas duas variáveis (AI_Supervision, Failure_Within_7_Days). A técnica de *Cast* consistem em converter uma variável de um tipo para outro (The Pandas Development Team, 2024). variáveis do tipo booleano tem o valor *True* convertido para 1 e *False* convertido para 0.

Para transforma as variáveis do tipo *object* (Machine_ID, Machine_Type) em um tipo aceitável para o modelo de ML, foi analisado algumas estratégias de conversão. A tabela 6 mostra o comparativo entre as técnicas *One-Hot Encode*, *Label Encode* e *Ordinal Encode*.

Tabela 6 – Comparativo entre técnicas de conversão de variáveis categóricas

Técnica	**Funcionamento**	**Vantagem**	**Desvantagem**
One-Hot Encoding	Cria uma coluna binária para cada categoria única.	Evita hierarquia falsa entre categorias nominais.	Aumenta a dimensionalidade (maior uso de memória).
Label Encoding	Atribui um inteiro único (0, 1, 2...) para cada categoria.	Simples e não altera a dimensão do conjunto de dados.	Pode introduzir uma ordem matemática inexistente.
Ordinal Encoding	Atribui inteiros seguindo uma ordem lógica pré-definida.	Preserva a relação de magnitude entre os dados.	Requer conhecimento prévio do especialista sobre a ordem.

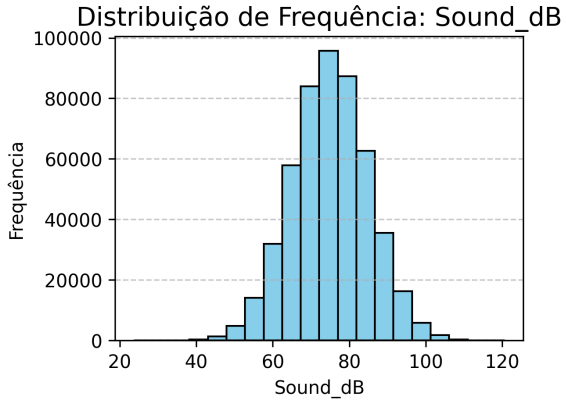
Fonte: Do autor baseado em: (ZHENG; CASARI, 2018)

Levando em conta que as duas variáveis categóricas são para identificação das máquinas e não exigem que haja uma hierarquia entre cada máquina, a técnica escolhida foi o *Label Encode*, por sua economia de mémoria, crucial em um MCU. Essa técnica atribui um valor numérico sequencial para cada uma das categorias. Apesar de ser simples, está écnica resolve o problema da categoria e não gera novas colunas de variáveis, como o *One-Hot Encode*. Por fim, duas variáveis foram descartadas. A variável Machine_ID é apenas uma variável de identificação individual de cada máquina, não apresentando relevância para o trabalho. Outra variável descartada foi Remaining_Useful_Life_days

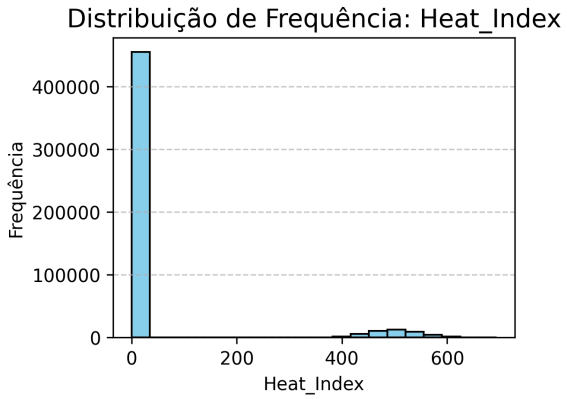
que é uma variável alvo usada em modelos de regressão, que não é o foco do trabalho.

4.4.3 Distribuição das variáveis

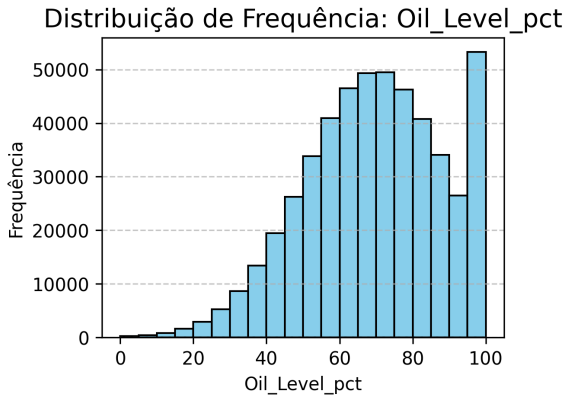
Antes da aplicação de qualquer técnica de aprendizado de máquina, é preciso compreender a distribuição das variáveis que compõem o conjunto de dados. Em sistemas de monitoramento industrial, cada sensor gera uma distribuição de valores que reflete o estado operacional do equipamento. Abaixo, as Figuras 4, 5 e 6 demosntram as distribuições das variáveis do conjunto de dados.



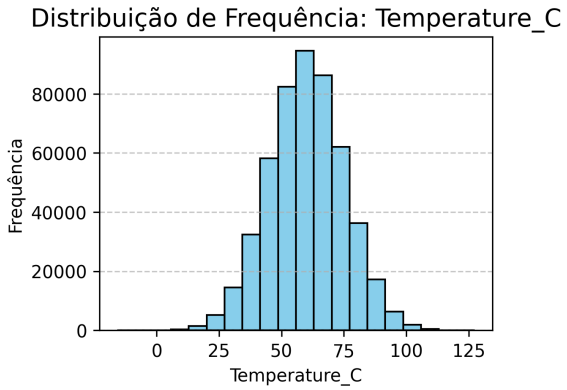
(a) Distribuição de *Sound_dB*



(b) Distribuição de *Heat_Index*



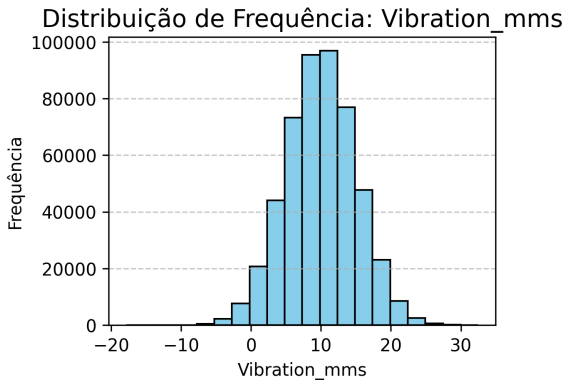
(c) Distribuição de *Oil_Level_pct*



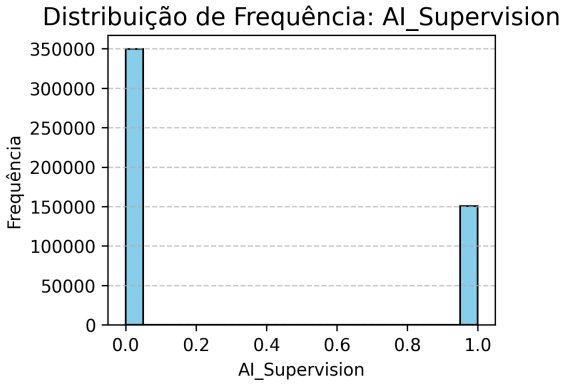
(d) Distribuição de *Temperature_C*

Figura 4 – Distribuições das variáveis do conjunto de dados (Parte 1 de 3)

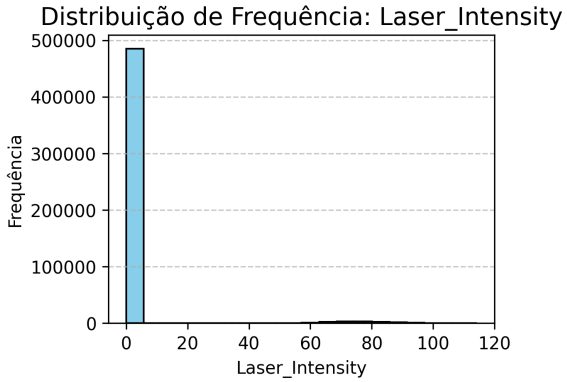
Fonte: Do autor



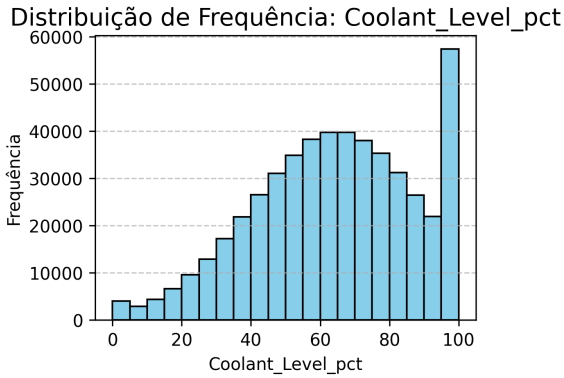
(a) Distribuição de *Vibration_mms*



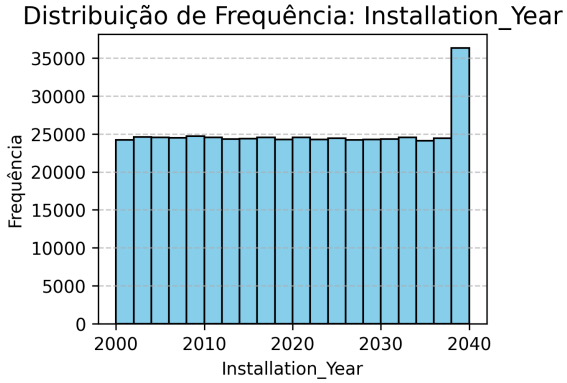
(b) Distribuição de *AI_Supervision*



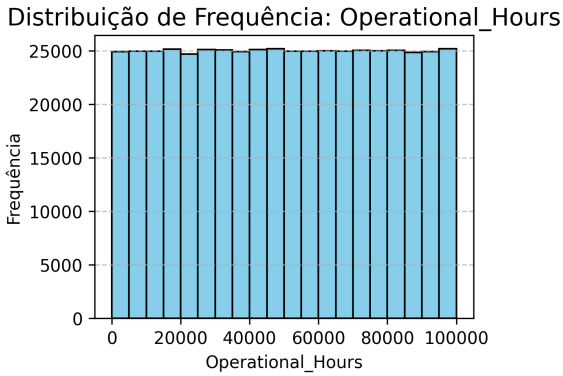
(c) Distribuição de *Laser_Intensity*



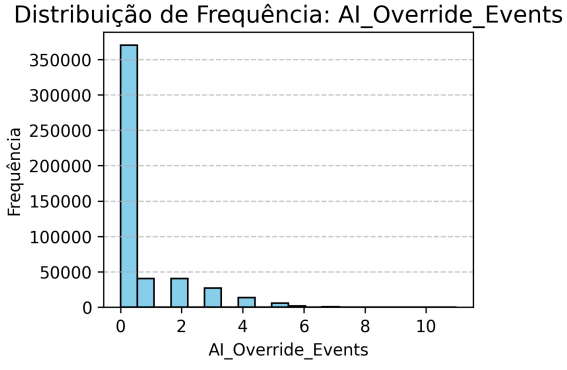
(d) Distribuição de *Coolant_Level_pct*



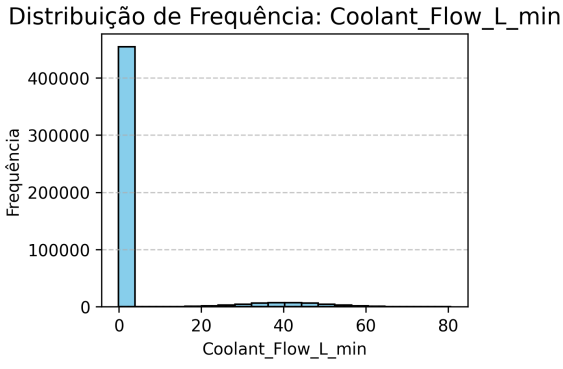
(e) Distribuição de *Installation_Year*



(f) Distribuição de *Operational_Hours*



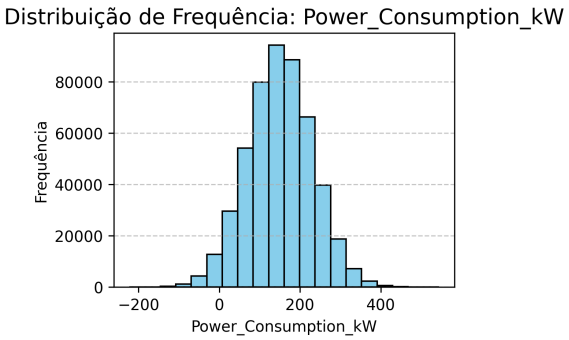
(g) Distribuição de *AI_Override_Events*



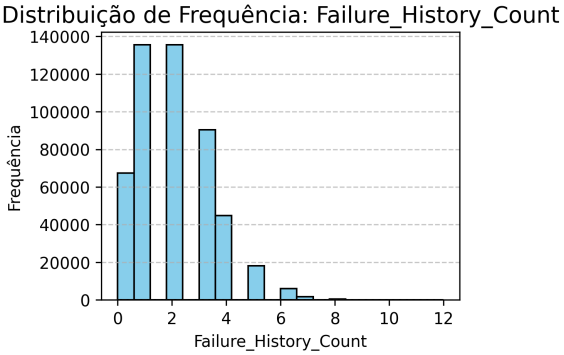
(h) Distribuição de *Coolant_Flow_L_min*

Figura 5 – Distribuições das variáveis do conjunto de dados (Parte 2 de 3)

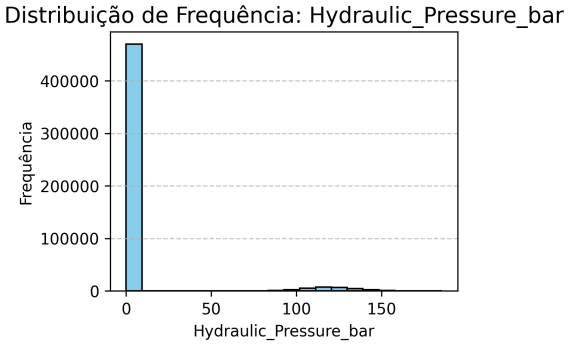
Fonte: Do autor



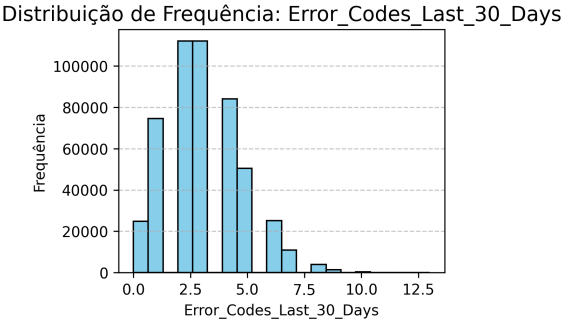
(a) Distribuição de *Power_Consumption_kW*



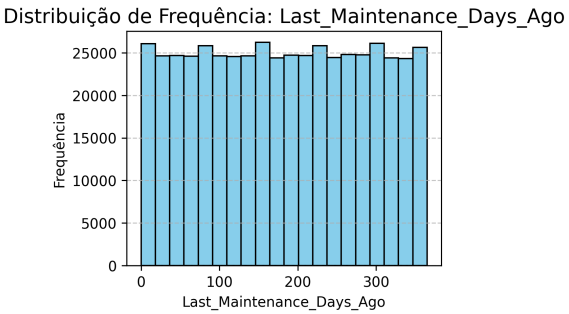
(b) Distribuição de *Failure_History_Count*



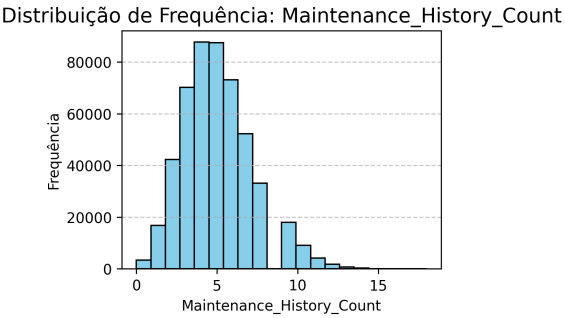
(c) Distribuição de *Hydraulic_Pressure_bar*



(d) Distribuição de *Error_Codes_Last_30_Days*



(e) Distribuição de *Last_Maintenance_Days_Ago*



(f) Distribuição de *Maintenance_History_Count*

Figura 6 – Distribuições das variáveis do conjunto de dados (Parte 3 de 3)

Fonte: Do autor

A análise da distribuição das variáveis é um pré-requisito para o emprego do classificador *Naive Bayes*, visto que o modelo fundamenta-se na suposição de que as variáveis seguem uma distribuição normal. Observando os histogramas gerados, percebe-se que variáveis como Temperature_C, Sound_dB, Oil_Level_pct, Vibration_mms, Coolant_Level_pct, Power_Consumption_kW, Failure_History_Count, Maintenance_History_Count, Failure_History_Count e Error_Codes_Last_30_Days exibem uma tendência central clara, validando a abordagem probabilística. Embora o Naive Bayes assuma a gaussianidade, e as demais variáveis não sigam uma curva normal, o modelo

é conhecido na literatura como um classificador robusto que mantém boa performance mesmo com desvios da normalidade (ZHANG, 2004), especialmente após o balanceamento de classes explicado na Seção 4.6.

4.5 Seleção de características

Pensando na limitação do ambienta na qual o modelo de ML irá ser executado, principalmente na limitação de armazenamento e de processamento, é importante estabelecer quais são as características que devem ser utilizadas durante a classificação dos dados. A metodologia adotada neste trabalho utiliza uma abordagem sequencial composta pelo algoritmo Random Forest para seleção de características, seguido pela PCA para interpretação da variância estrutural.

Segundo Breiman (2001), o uso inicial do Random Forest justifica-se pela sua capacidade intrínseca de medir a importância das variáveis, baseando-se no critério de impureza de Gini ou na precisão de permutação. Diferente de métodos lineares, o Random Forest identifica variáveis que possuem forte poder preditivo mesmo em relações não lineares complexas (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). Esta etapa atua como um filtro de ruído, que elimina atributos irrelevantes antes da transformação dimensional.

Após a redução do conjunto original para as variáveis mais impactantes, a aplicação do PCA permite observar como essas características se organizam e se correlacionam nos componentes principais (JOLLIFFE; CADIMA, 2016). Enquanto o Random Forest determina o que é importante para a classificação de falhas, o PCA revela o como essas variáveis explicam a variabilidade do sistema, permitindo identificar boas variáveis para uso no modelo. O resultado do PCA foi:

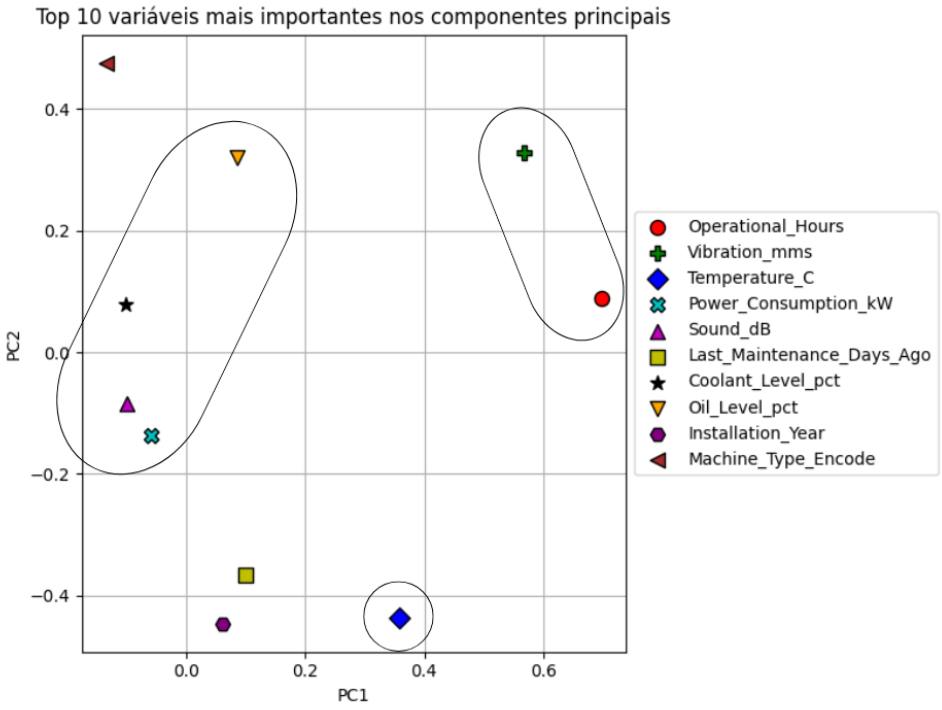


Figura 7 – Análise de PCA

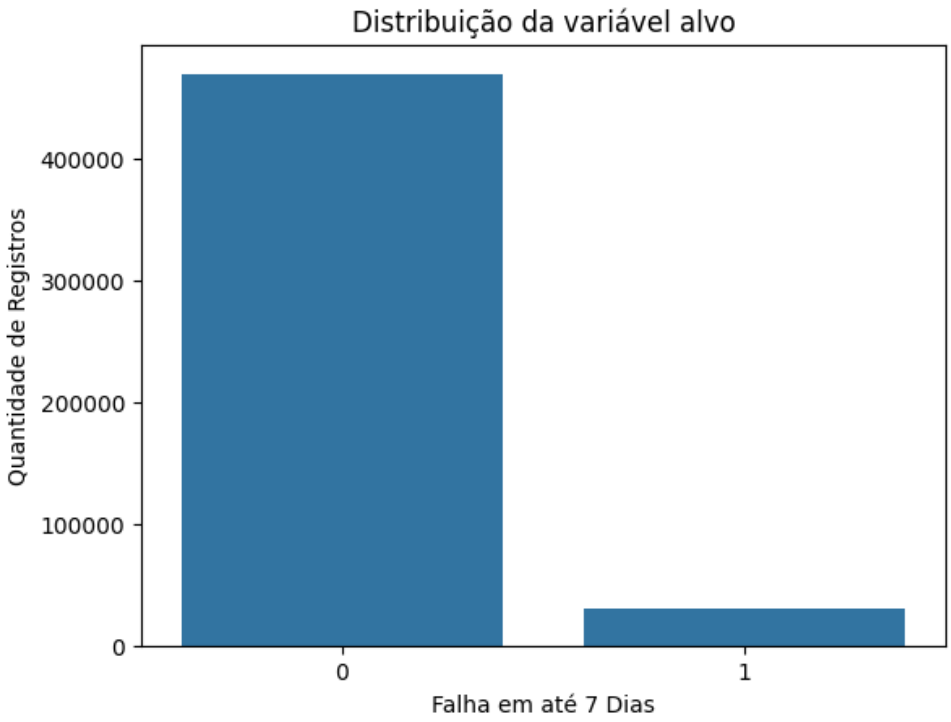
Fonte: Do autor

Ao analisar o resultado que pode ser visualizado na Figura 7, 7 das 10 variáveis mais importantes para os componentes são dimensões que podem serem obtidas através de sensores conectados ao MCU. Isso torna o problema mais fácil de ser generalizado e facilita a coleta de informações das máquinas e equipamentos indutriais. Outro ponto de destaque é a variável *Machine_Type_Encode* que representa o tipo de máquina que aqueles dados pertecem. Isso abre a possibilidade de subdividar a base completa em subconjuntos que representam apenas os dados de uma maquína ou equipamento industrial, especializando o MCU em detectar falhas da máquina, ou máquinas de mesmo tipo, que ele está conectado.

4.6 Balanceamento do conjunto de dados

Primeiro é importante saber a quantidade de registros que há da variável principal de classificação. Para isso, foi feito uma contagem da quantidade de cada um dos dados e realizado a construção de um gráfico de barras que demonstrasse isso.

Figura 8 – Distribuição da variável principal



Fonte: Do autor

Como é possível verificar no gráfico da Figura 8, há mais registros de dados de operação normal (0), 469.968 registros, do que de dados de operação que leve a falha (1), 30.032 registros. dessa forma, é necessário realizar uma etapa de balanceamento da base de dados para evitar que o algoritmo treinado fique enviesado.

Segundo Zheng et al. (2020) natureza dos sistemas de monitoramento industrial para manutenção preditiva frequentemente resulta em conjuntos de dados altamente desbalanceados, onde a classe de operação normal prevalece amplamente sobre os eventos de falha. Para mitigar o viés algorítmico em direção à classe majoritária, o que poderia levar a uma alta acurácia, porém com baixa capacidade de detecção de falhas reais (Recall), optou-se pela estratégia de *Random Undersumpling* com proporção 1 para 1.

A escolha por extrair uma quantidade de exemplos de operação plena idêntica ao número de eventos de falha registrados para cada máquina e de subdividir o conjunto de dados em subconjuntos com dados de cada máquina especificamente se justifica por:

1. **Integridade estatística:** Ao utilizar apenas dados reais de operação, evita-se a introdução de ruídos sintéticos ou padrões artificiais que poderiam surgir com técnicas de sobreamostragem como o SMOTE (CHAWLA et al., 2002).

2. **Equidade de classe:** O balanceamento 1:1 garante que o classificador Naive Bayes atribua pesos idênticos às probabilidades, otimizando a fronteira de decisão para a

identificação de anomalias (HE; GARCIA, 2009). Além de amenizar o fato de que a distribuição das variáveis não segue uma curva normal.

3. **Eficiência em TinyML:** Dado que o objetivo final é a implementação em sistemas embarcados com restrições de memória, treinar o modelo em um subconjunto enxuto e equilibrado foca a aprendizagem nos vetores de características mais representativos de cada estado da máquina (SÁNCHEZ; BATISTA; RUIZ-SHULCLOPER, 2003).

Para compensar o caráter "alarmista" intrínseco de modelos treinados em bases balanceadas aplicados a cenários reais, onde falhas são raras (ZHENG et al., 2020), implementou-se um ajuste no limiar de decisão de probabilidade (*Threshold*) para 0,8, O que será melhor explicado na Seção 4.7. Essa medida visa aumentar a precisão do sistema e reduzir o custo operacional gerado por falsos positivos.

4.6.1 Setup de treinamento

Para o treinamento embarcado do modelo, foram gerados 10 subconjuntos de dados para cada uma das 30 máquinas avaliadas. Cada subconjunto foi estruturado por meio da estratégia de *Random Undersampling* e, posteriormente, particionado em amostras de treino e validação, nas proporções de 80% e 20%, respectivamente. Os dados de treinamento foram passados para o MCU via cabo USB pelo terminal de um computador, e os resultados após o treinamento e validação embarcados também eram imprimidos no terminal do computador.

4.7 Detalhes da implementação

Para a detecção de anomalias, foi desenvolvido um *Script* em linguagem C++ para o MCU ESP32, implementando o algoritmo *Naive Bayes*. O sistema foi decomposto em quatro blocos lógicos fundamentais para garantir a portabilidade e eficiência em sistemas embarcados. Segue os pseudocódigos de cada bloco:

Algoritmo 1: Cálculo da verossimilhança Gaussiana

1 **Função** `calculateLikelihood`(x, μ, σ^2)`:`
2 $\quad exp \leftarrow \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$
3 $\quad$ **return** $\frac{1}{\sqrt{2\pi\sigma^2}} \cdot exp$

Algoritmo 2: Fase de treinamento estatístico

Função `trainModel(`*DadosTreino, Rótulos*`):`
- Zerar médias e variâncias para cada classe
- **for** *cada amostra* i **do**
  - Classe $c \leftarrow$ Rótulos$[i]$
  - $\text{SomaMédias}_c \leftarrow \text{SomaMédias}_c + \text{DadosTreino}[i]$
- **for** *cada classe* c **do**
  - $\mu_c \leftarrow \text{SomaMédias}_c / \text{N}_c$
  - $\text{Prior}_c \leftarrow \text{N}_c / \text{Total}$
- **for** *cada amostra* i **do**
  - $c \leftarrow$ Rótulos$[i]$
  - $\sigma_c^2 \leftarrow \sigma_c^2 + (\text{DadosTreino}[i] - \mu_c)^2$
- $\sigma_c^2 \leftarrow (\sigma_c^2 / \text{N}_c) + \epsilon$

Algoritmo 3: Inferência e tomada de decisão

Função `predict(`*entrada*`):`
- **for** *cada classe* c **do**
  - $\text{log\_post}_c \leftarrow \log(\text{Prior}_c)$
  - **for** *cada característica* f **do**
    - $L \leftarrow \text{calculateLikelihood}(\text{entrada}_f, \mu_{c,f}, \sigma_{c,f}^2)$
    - $\text{log\_post}_c \leftarrow \text{log\_post}_c + \log(L + 10^{-9})$
- $\text{prob} \leftarrow \text{Softmax}(\text{log\_post}_1)$
- **return** $\text{prob} \geq \text{Threshold}?1 : 0$

Algoritmo 4: Setup e avaliação de desempenho

```
1  Função setup/Avaliação(DadosTeste):
2      trainModel(DadosTreino, Rótulos)
3      for cada amostra de teste j do
4          pred ← predict(amostra_j)
5          real ← test_y_true[j]
6          if pred == 1 e real == 1 then
7              TP ← TP + 1
8          else if pred == 0 e real == 0 then
9              TN ← TN + 1
10         else if pred == 1 e real == 0 then
11             FP ← FP + 1
12         else if pred == 0 e real == 1 then
13             FN ← FN + 1
14     Acurácia ← (TP+TN) / Total de Amostras
15     Precisão ← TP / (TP+FP)
16     Recall ← TP / (TP+FN)
17     F1-Score ← 2 · (Precisão·Recall) / (Precisão+Recall)
```

Implementa a função de densidade de probabilidade da Distribuição Gaussiana. A função `calculateLikelihood()` calcula a probabilidade de um valor de sensor específico pertencer a uma determinada classe (Normal ou Falha), assumindo que os dados seguem uma curva normal. O resultado desse calculo é utilizado na função `trainModel()`, que realiza o aprendizado estatístico a partir do conjunto de dados. Em vez de armazenar amostras brutas, o que esgotaria a memória do MCU, o modelo resume os dados em três parâmetros fundamentais por classe: a média (μ), a variância (σ^2) e a probabilidade *a priori*. A inclusão de um fator ϵ ($1e-6$) na variância é uma técnica de suavização para evitar divisões por zero durante a inferência.

Após o treinamento, a função `predict()` é executada para realizar as inferências. Para garantir estabilidade numérica em arquiteturas de 32 bits, como o ESP32, a função utiliza a soma de logaritmos em vez do produto de probabilidades, prevenindo o problema de *underflow* aritmético. Visando reduzir o caráter "alarmista" do modelo - Como dito na Seção 4.6 - e minimizar o custo gerado por falsos positivos na manutenção preditiva, implementou-se um limiar (*threshold*) customizado de 0,8. Assim, o sistema só classifica um evento como "Falha" caso a probabilidade calculada supere 80%. Caso contrário, a amostra é classificada como "Operação Normal", priorizando a continuidade da produção em cenários de incerteza moderada. Esse valor foi determinado testando diversos valores de 10% em 10% de aumento gradativo até encontrar um ponto de equilíbrio na qual as

métricas de avaliação sejam positivas.

Por fim, a função `setup()`, além de inicializar o hardware e iniciar o processo de treinamento, executa a validação do modelo. São calculadas as métricas de desempenho Acurácia, Precisão, *Recall* e *F1-Score*, fornecendo uma visão quantitativa da capacidade do modelo em identificar falhas reais.

4.8 Considerações finais

Este capítulo definiu o MCU, o modelo de ML e o conjunto de dados que vão ser utilizados no treinamento embarcado. Além disso, foram feitos tratamentos importantes na base de dados antes do treinamento e foi demonstrado o desenvolvimento de um *script* que aplica o modelo escolhido no MCU.

5 Resultados

Nesta seção, são apresentados e discutidos os resultados obtidos no treinamento embarcado do modelo. Os experimentos foram realizados em 30 máquinas distintas, utilizando a estratégia de *Random Undersampling* para o balanceamento dos dados. Conforme descrito no capítulo 4, cada máquina foi submetida a 10 rodadas de testes, sendo os valores reportados a seguir as médias aritméticas das métricas de desempenho observadas nessas repetições.

A Tabela 7 consolida os resultados de Acurácia, Precisão, *Recall* e *F1-Score* para cada uma das unidades avaliadas.

Tabela 7 – Média dos resultados de classificação após 10 testes por máquina.

Máquina	**Acurácia**	**Precisão**	**Recall**	**F1-Score**
M0	0,93	0,91	0,95	0,93
M1	0,91	0,91	0,92	0,91
M2	0,94	0,94	0,95	0,94
M3	0,93	0,93	0,94	0,93
M4	0,96	0,95	0,97	0,96
M5	0,96	0,94	0,98	0,96
M6	0,94	0,96	0,92	0,94
M7	0,96	0,96	0,97	0,97
M8	0,94	0,93	0,96	0,95
M9	0,94	0,92	0,96	0,94
M10	0,92	0,93	0,92	0,93
M11	0,96	0,96	0,97	0,97
M12	0,93	0,92	0,94	0,93
M13	0,93	0,93	0,93	0,93
M14	0,96	0,95	0,97	0,96
M15	0,95	0,94	0,95	0,95
M16	0,96	0,96	0,96	0,96
M17	0,94	0,93	0,94	0,94
M18	0,93	0,93	0,94	0,94
M19	0,94	0,94	0,94	0,94
M20	0,93	0,92	0,95	0,93
M21	0,94	0,96	0,94	0,95
M22	0,96	0,94	0,97	0,95
M23	0,93	0,95	0,91	0,93
M24	0,95	0,96	0,94	0,95
M25	0,94	0,95	0,93	0,94
M26	0,95	0,95	0,95	0,95
M27	0,92	0,90	0,95	0,92
M28	0,95	0,95	0,95	0,95
M29	0,95	0,95	0,94	0,95

Ao examinar os dados consolidados na Tabela 7, verifica-se um desempenho de classificação altamente consistente entre as 30 máquinas avaliadas. A Acurácia média apresentou uma oscilação reduzida, compreendida no intervalo entre 91% (Máquina M1) e

96% (máquinas M4, M7 e M11), o que evidencia uma elevada capacidade de generalização do modelo diante de diferentes fontes de dados.

Essa estabilidade métrica é diretamente atribuída à estratégia de partição de dados adotada. A utilização do limiar de 80% para o conjunto de treinamento mostrou-se determinante para o sucesso do aprendizado, uma vez que permitiu ao algoritmo capturar de forma abrangente as características intrínsecas de cada classe, mesmo após a redução volumétrica imposta pelo *Random Undersampling*. Esse volume substancial de dados de treino garantiu que o modelo convergisse para parâmetros otimizados, refletindo-se em métricas de validação superiores e estáveis.

No que diz respeito ao equilíbrio entre Precisão e *Recall*, observa-se um cenário ideal para aplicações de monitoramento industrial. O destaque para a Máquina M5, que atingiu um *Recall* de 98%, demonstra a eficácia do sistema em identificar quase a totalidade das ocorrências positivas, fator crítico para a minimização de falsos negativos que poderiam resultar em falhas catastróficas não detectadas.

O limiar de 80% foi aplicado à saída probabilística do modelo, definindo que uma falha só é reportada caso a probabilidade estimada supere esse patamar. Essa estratégia foi fundamental para reduzir a característica alarmista do modelo, assegurando que apenas condições de alta criticidade resultem em alertas, o que é essencial para evitar falsos positivos em ambientes industriais.

A aplicação desse limiar rigoroso reflete-se na consistência da *Precisão*, que se manteve igual ou superior a 90% em todas as máquinas avaliadas. Na Máquina M11, por exemplo, alcançou-se uma precisão de 96%, o que demonstra que, ao elevar a exigência para a classificação de falhas, o modelo tornou-se extremamente assertivo em suas predições.

Apesar da adoção de um critério de decisão conservador, o modelo não apresentou perda significativa de sensibilidade. O *Recall* médio atingiu valores expressivos, como 98% na Máquina M5 e 97% nas Máquinas M7 e M14, indicando que a base de treinamento (80% dos dados) permitiu ao algoritmo aprender os padrões de falha de forma tão sólida que mesmo um limiar de 80% de probabilidade não impediu a detecção da vasta maioria dos eventos reais.

A análise das 30 máquinas mostra que a combinação do *Random Undersampling* com o limiar de decisão de 80% resultou em *F1-Scores* estáveis, predominantemente iguais ou superiores a 93%. Isso comprova que a redução da característica alarmista não comprometeu o equilíbrio global do classificador, validando a viabilidade do treinamento embarcado para diagnósticos precisos.

Ao comparar os dados com o estudo de Seidel et al. (2024), observa-se uma superioridade significativa no desempenho da inferência embarcada. Enquanto o referido autor obteve acurácias entre 71,5% e 83,3% na execução em microcontrolador, o presente

trabalho alcançou níveis de acurácia média entre 91% e 96% em todas as 30 máquinas avaliadas. Essa disparidade é notável considerando que este trabalho utiliza o classificador Naive Bayes, que é computacionalmente mais "leve"e possui menor pegada de memória do que os modelos de Árvore de Decisão e SVM utilizados por Seidel et al. (2024).

Relação Custo-Benefício e viabilidade de Hardware em relação ao trabalho de Franco (2020), a comparação foca na viabilidade econômica e na restrição de hardware. O autor utilizou uma Raspberry Pi 3, um dispositivo com alto poder de processamento mas com custo de mercado elevado (aproximadamente R$ 400,00). Em contrapartida, esta pesquisa validou o uso da linha ESP32, um microcontrolador de baixo custo e alta disponibilidade.

Mesmo operando em um ecossistema de hardware muito mais restrito e econômico que a Raspberry Pi 3, os resultados não apresentaram degradação de performance. Pelo contrário, o sistema manteve índices de *F1-Score* predominantemente iguais ou superiores a 93%, com a vantagem adicional da redução do caráter alarmista do modelo devido ao limiar de probabilidade implementado. Isso prova que é possível democratizar a manutenção preditiva em pequenas indústrias utilizando componentes de baixo custo sem abdicar da precisão diagnóstica.

6 Conclusão

Este trabalho demonstrou a viabilidade técnica e a eficiência da aplicação de sistemas de monitoramento na borda para a manutenção preditiva industrial. A utilização do microcontrolador ESP32 consolidou-se como uma escolha estratégica, oferecendo o suporte necessário para o processamento local de dados com baixo custo e alta disponibilidade. O modelo classificador Naive Bayes, selecionado por sua leveza computacional, provou ser altamente eficaz ao operar em um ambiente de hardware restrito, garantindo respostas rápidas sem a dependência de processamento em nuvem.

Os resultados obtidos foram expressivos e validam a robustez da solução proposta em diversas condições operacionais. Nas inferências realizadas em 30 máquinas distintas, observou-se uma acurácia consistente entre 91% e 96%, demonstrando uma elevada capacidade de generalização do sistema. Além disso, o desempenho alcançou níveis de sensibilidade notáveis, com o recall atingindo 98% em unidades específicas e o F1-Score mantendo-se igual ou superior a 93%. A precisão do sistema também apresentou alto desempenho, alcançando picos de 96% em diversas unidades avaliadas. Esses indicadores quantificam o sucesso do treinamento embarcado e confirmam que a simplificação do algoritmo não resultou em perda de confiabilidade diagnóstica.

Em suma, a pesquisa atingiu seus objetivos ao integrar hardware de baixo custo e inteligência artificial para a detecção de falhas industriais. A estabilidade das métricas apresentadas reforça que o sistema está apto para atuar como uma ferramenta de suporte à decisão em ambientes insdustriais reais. No entanto, o desenvolvimento tecnológico na área sugere que o aprimoramento contínuo é fundamental, indicando que ainda há espaço para a realização de trabalhos futuros que expandam as capacidades de coleta e processamento do sistema implementado.

6.1 Trabalhos futuros

Como trabalhos futuros, seria interessante realizar as seguintes tarefas:

- Realizar inferências embarcadas utilizando um volume superior aos 10 subconjuntos de dados atuais por máquina, visando aumentar a robustez estatística e a generalização dos resultados obtidos.
- Construir uma base de dados proprietária com informações extraídas diretamente de ativos industriais em ambiente real, contemplando tanto o regime de plena operação quanto o histórico de falhas mecânicas e elétricas.

- Adaptar o *script* implementado para explorar a arquitetura de dois núcleos do microcontrolador, destinando um núcleo exclusivamente à aquisição e tratamento de sinais e o segundo à execução das inferências de ML.
- Utilizar instruções específicas de otimização de hardware e bibliotecas de baixo nível para potencializar as operações aritméticas do algoritmo.
- Modificar o código para MicroPython, possibilitando o uso de bibliotecas implementadas, como Tensorflow Lite.
- Executar inferência contínua com dados capturados em tempo real, comparando a capacidade de predição imediata do sistema com o comportamento observado nos equipamentos.

Referências

ABRAMAN. *Documento Nacional de Manutenção e Gestão de Ativos*. Rio de Janeiro, 2021. Citado 2 vezes nas páginas 17 e 18.

ARDUINO SA. *Arduino Nano Every Datasheet*. Chiasso, Suíça, 2024. Disponível em: <https://docs.arduino.cc/resources/datasheets/ABX00028-datasheet.pdf>. Citado na página 37.

ASHTON, K. That 'internet of things' thing. *RFID journal*, v. 22, n. 7, p. 97–114, 2009. Citado na página 22.

BISHOP, C. M. *Pattern Recognition and Machine Learning*. New York: Springer, 2006. Citado 2 vezes nas páginas 28 e 39.

BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001. Citado 2 vezes nas páginas 28 e 49.

Can Özensoy. *Industrial IoT Dataset (Synthetic)*. 2025. Disponível em: <https://www.kaggle.com/datasets/canozensoy/industrial-iot-dataset-synthetic>. Acesso em: 18 mar. 2025. Citado na página 40.

CHAWLA, N. V. Data mining for imbalanced datasets: An overview. In: *Data Mining and Knowledge Discovery Handbook*. Boston, MA: Springer, 2005. p. 853–867. Citado na página 30.

CHAWLA, N. V. et al. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, v. 16, p. 321–357, 2002. Citado na página 51.

CNI. *Investimentos e Produtividade na Indústria Brasileira*. Brasília, 2018 – 2022. Citado 2 vezes nas páginas 17 e 18.

CORTES, C.; VAPNIK, V. Support-vector networks. *Machine Learning*, Springer, v. 20, n. 3, p. 273–297, 1995. Citado na página 39.

COVER, T.; HART, P. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, IEEE, v. 13, n. 1, p. 21–27, 1967. Citado na página 39.

DANG, N.-N. *Embedded Systems: Architecture, Programming and Design*. [S.l.]: CRC Press, 2022. Citado 3 vezes nas páginas 9, 22 e 23.

DUTTA, L.; BHARALI, S. Tinyml: A step towards next-generation edge ai. *IEEE Access*, IEEE, v. 9, p. 23126–23139, 2020. Citado 2 vezes nas páginas 16 e 18.

Espressif Systems. *ESP32 Series Datasheet*. Shanghai, China, 2024. Acesso em: 14 dez. 2025. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>. Citado 2 vezes nas páginas 36 e 37.

FRANCO, I. T. Manutenção preditiva utilizando técnicas de machine learning em um sistema embarcado. Universidade do Vale do Rio dos Sinos, 2020. Citado 3 vezes nas páginas 33, 34 e 58.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning*. [S.l.]: MIT press, 2016. Citado na página 40.

GRAFF, R. S. Sistema industrial de internet das coisas para predição de anomalias aplicando técnicas de aprendizado de máquina. Universidade do Vale do Rio dos Sinos, 2019. Citado na página 34.

HAND, D. J.; YU, K. Idiot's bayes: not so stupid after all? *International Statistical Review*, Wiley, v. 69, n. 3, p. 385–398, 2001. Citado 2 vezes nas páginas 38 e 39.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2. ed. New York: Springer, 2009. Citado 4 vezes nas páginas 27, 28, 29 e 49.

HE, H.; GARCIA, E. A. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 21, n. 9, p. 1263–1284, 2009. Citado 3 vezes nas páginas 30, 31 e 52.

HERMANN, M.; PENTEK, T.; OTTO, B. Design principles for industrie 4.0 scenarios. In: *51st Hawaii International Conference on System Sciences*. [S.l.: s.n.], 2016. Citado na página 21.

HOTELLING, H. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, Warwick & York, v. 24, n. 6, p. 417, 1933. Citado na página 30.

JOLLIFFE, I. T. *Principal Component Analysis*. New York: Springer-Verlag, 2002. Citado na página 29.

JOLLIFFE, I. T.; CADIMA, J. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, The Royal Society, v. 374, n. 2067, p. 20150202, 2016. Citado na página 49.

JORDAN, M. I.; MITCHELL, T. M. Machine learning: Trends, perspectives, and prospects. *Science*, v. 349, n. 6245, p. 255–260, 2015. Citado na página 40.

KAGERMANN, H. et al. Recommendations for implementing the strategic initiative industrie 4.0. *Acatech*, 2013. Citado na página 21.

Kaggle. *Kaggle: Your Machine Learning and Data Science Community*. 2025. Disponível em: <https://www.kaggle.com>. Acesso em: 15 mar. 2025. Citado na página 40.

KARDEC, A.; NASCIF, J. *Gestão de Manutenção e Terceirização: uma abordagem estratégica*. 3. ed. Rio de Janeiro: Qualitymark, 2009. Citado na página 24.

LASI, H. et al. Industry 4.0. *Business & Information Systems Engineering*, v. 6, p. 239–242, 2014. Citado na página 21.

LEE, J.; BAGHERI, B.; KAO, H.-A. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters*, Elsevier, v. 3, p. 18–23, 2014. Citado 5 vezes nas páginas 16, 23, 24, 26 e 27.

MARWEDEL, P. *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things*. 4th. ed. [S.l.]: Springer Nature, 2021. Citado na página 23.

MITCHELL, T. M. *Machine Learning*. New York: McGraw-Hill, 1997. Citado 2 vezes nas páginas 27 e 38.

MOBLEY, R. K. *An Introduction to Predictive Maintenance*. 2. ed. Amsterdam: Elsevier, 2002. Citado 5 vezes nas páginas 16, 18, 23, 24 e 25.

MONOSTORI, L. Cyber-physical production systems: Roots, expectations and r&d challenges. *Procedia CIRP*, v. 17, p. 9–13, 2014. Citado na página 16.

PORTER, M. E.; HEPPELMANN, J. E. How smart, connected products are transforming competition. *Harvard Business Review*, v. 92, n. 11, p. 64–88, 2014. Citado na página 22.

POWERS, D. M. W. Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, v. 2, n. 1, p. 37–63, 2011. Citado na página 31.

RANDALL, R. B. *Vibration-based Condition Monitoring: Industrial, Aerospace and Automotive Applications*. Chichester: John Wiley & Sons, 2011. Citado 3 vezes nas páginas 23, 24 e 25.

Raspberry Pi Foundation. *Raspberry Pi 3 Model B*. 2026. <https://www.raspberrypi.com/products/raspberry-pi-3-model-b/>. Acessado em: 03 de fevereiro de 2026. Citado 2 vezes nas páginas 34 e 35.

RASPBERRY PI LTD. *RP2040 Datasheet: A microcontroller by Raspberry Pi*. Cambridge, Reino Unido, 2024. Disponível em: <https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf>. Citado na página 37.

RAY, P. P. A review on tinyml: State-of-the-art and prospects. *Journal of King Saud University - Computer and Information Sciences*, 2021. Citado 2 vezes nas páginas 16 e 18.

RÜSSMANN, M. et al. Industry 4.0: The future of productivity and growth in manufacturing industries. *Boston Consulting Group*, 2015. Citado na página 22.

SÁNCHEZ, J. S.; BATISTA, G. E.; RUIZ-SHULCLOPER, J. Analysis of data filtering techniques in a class imbalance problem. In: *Conference of the Spanish Association for Artificial Intelligence*. [S.l.: s.n.], 2003. p. 503–510. Citado na página 52.

SATYANARAYANAN, M. The emergence of edge computing. *Computer*, IEEE, v. 50, n. 1, p. 30–39, 2017. Citado na página 25.

SCHWAB, K. *The Fourth Industrial Revolution*. [S.l.]: World Economic Forum, 2016. Citado na página 21.

SEIDEL, F. et al. Detecção de falhas mecânicas em motores elétricos baseada em modelos de aprendizado de máquina embarcados em microcontrolador. 2024. Citado 3 vezes nas páginas 33, 57 e 58.

SHANNON, C. E. A mathematical theory of communication. *The Bell System Technical Journal*, 1948. Citado na página 22.

SHI, W. et al. Edge computing: Vision and challenges. *IEEE internet of things journal*, IEEE, v. 3, n. 5, p. 637–646, 2016. Citado na página 25.

STMICROELECTRONICS. *RM0383 Reference manual: STM32F411xC/E advanced Arm®-based 32-bit MCUs*. Genebra, Suíça, 2025. Disponível em: <https://www.st.com/resource/en/reference_manual/rm0383-stm32f411xce-advanced-armbased-32bit-mcus-stmicroelectronics.pdf>. Citado na página 37.

The Pandas Development Team. *pandas.DataFrame.astype*. [S.l.], 2024. Acessado em: 7 de fev. 2026. Disponível em: <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.astype.html>. Citado na página 45.

UCI Machine Learning Repository. *Home*. 2025. Disponível em: <https://archive.ics.uci.edu>. Acesso em: 16 mar. 2025. Citado na página 40.

WARDEN, P.; SITUNAYAKE, D. *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. Sebastopol: O'Reilly Media, 2019. Citado 5 vezes nas páginas 16, 17, 18, 24 e 26.

ZHANG, H. The optimality of naive bayes. In: *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference*. [S.l.: s.n.], 2004. p. 562–567. Citado 2 vezes nas páginas 38 e 49.

ZHANG, W.; YANG, D.; WANG, H. Deep learning for smart manufacturing: Methods and applications. *Journal of Manufacturing Systems*, Elsevier, v. 54, p. 255–276, 2020. Citado 2 vezes nas páginas 26 e 27.

ZHENG, A.; CASARI, A. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. Sebastopol: O'Reilly Media, 2018. Citado na página 45.

ZHENG, P. et al. Predictive maintenance in the era of industry 4.0. *Applied Sciences*, v. 10, n. 7, 2020. Citado 2 vezes nas páginas 51 e 52.