



Time-series Failure Prediction on Small Datasets Using Machine Learning

Caio B. S. Maior , and Thaylon G. Silva 

Abstract— Condition-based maintenance is a decision-making strategy using condition monitoring information to optimize the availability of operational plants. In this context, machine learning techniques are useful and have been used in predicting the remaining useful life (RUL) of equipment to ensure the overall safety and reliability of the system through maintenance policies and, consequently, reducing costs arising from the failure. These databases are not large which is tricky for data-driven models. In this study, we consider five different databases containing the failure times from distinct real-world equipment. Here, four different regression algorithms were compared for RUL prediction, namely: Support Vector Regression (SVR), Decision Tree (DT), Multilayer Perceptron (MLP) and K-Nearest Neighbors (KNN). Furthermore, aiming to improve the data quality, the Empirical Mode Decomposition (EMD) was used, which is responsible for pre-processing the input data used on the predictive modeling. We optimize the models' hyperparameters using grid-search cross-validation algorithm and the performance of each model is compared using the normalized root mean squared error (NRMSE). Considering the datasets analyzed, KNN model proves to be the most promising to perform the prognostic task in small datasets, adapting itself to the distinct characteristics of the different databases. In addition, we mention the better performance after optimizing the hyperparameters, which avoided overfitting problems and had low computational cost for the problems analyzed here.

Link to graphical and video abstracts, and to code: <https://latam.ieeer9.org/index.php/transactions/article/view/8296>

Index Terms— Machine Learning, Remaining Useful Life, Time-Series, Empirical Mode Decomposition, Predictive Models.

I. INTRODUÇÃO

Comumente, os sistemas produtivos são complexos, em linhas de produção integradas e compostas por vários componentes onde a falha de um equipamento impacta em perda de produtividade, horas extras de produção e queda da qualidade do produto. Neste contexto, a manutenção preditiva pode ser usada para monitorar os estados de saúde de equipamentos e planejar sua manutenção, evitando gastos desnecessários para a organização [1].

Ainda, sistemas críticos (ex., hospitais, usinas nucleares, aeronaves, poços de petróleo) exigem alta confiabilidade pois, além dos custos causados pela falha do ativo em si, seu funcionamento impacta diretamente a saúde e segurança das pessoas, podendo ocasionar em perdas imensuráveis [2]. Assim, uma acurada estimação do tempo de falha permite uma melhor tomada de decisão de procedimentos preventivos de manutenção ou de mitigação de consequências.

C. B. S. Maior and T.G. Silva are with Federal University of Pernambuco, Caruaru, Pernambuco, Brazil (e-mails: caio.maior@ufpe.br and thaylon.gomes@ufpe.br).

Neste cenário, Prognóstico e Gerenciamento de Saúde (PHM) é a área da engenharia responsável por avaliar, prognosticar, diagnosticar e gerenciar a saúde de sistemas, útil para ajudar na tomada de decisão da manutenção produzindo uma gestão mais eficaz [3], [4]. Aqui, o conceito de vida útil remanescente (RUL) está relacionado com a previsão da vida dos componentes, sendo formalmente definido como o tempo restante para que um sistema, máquina ou componente execute suas capacidades funcionais antes da falha [5].

Embora seja uma tarefa difícil, a estimação de RUL pode ser realizada a partir de diferentes abordagens. Por exemplo, abordagens Bayesianas baseadas em um modelo de regressão gaussiano são apresentadas por Zegarra, Vargas-Machuca e Coronado [6] para estimar o RUL de máquinas fresadoras, e por Tanwar e Raghavan [7] para a previsão da vida útil remanescente de óleo lubrificante. Entretanto, os pressupostos necessários para utilização de certos modelos matemáticos podem limitar a representação da realidade. Estas simplificações ocorrem tanto devido à falta de informações relevantes, ao desconhecimento das variáveis que efetivamente afetam o sistema, ou ainda devido à impossibilidade de tratabilidade matemática e numérica [8].

Alternativamente, técnicas baseadas em aprendizado de máquina (ML) também são comuns na estimação do RUL de equipamentos, com o benefício de não necessitar suposição prévia sobre os dados. De fato, comparação de desempenho de abordagens preditivas de ML com abordagens clássicas de previsão de séries temporais (e.g., ARIMA, Holt-Winter, decomposição) indicam resultados promissores para os modelos de ML, com estimativas precisas em abordagens generalistas [9], [10], [11].

Neste sentido, Tian, Wong e Safaei [12] usaram um modelo de rede neural artificial (RNA) para prever o RUL de mancais de bombas a partir de dados históricos de falha e censura, e sinais de vibração. Já Ordóñez *et al.* [13] desenvolveram um modelo ARIMA integrado a um modelo de *Support Vector Machine* (SVM) para previsão de RUL em motores de aeronaves. García Nieto *et al.* [14] também desenvolveram um modelo híbrido para prever o RUL de motores de aeronaves, porém considerando uma otimização de parâmetros do algoritmo do SVM a partir de *Particle Swarm Optimization* (PSO). Ideia similar foi seguida por Maior *et al.* [1], [8] no qual o PSO-SVM é utilizado para calcular RUL a partir de dados de falha e dados de vibração, respectivamente. Já Wang *et al.* [15] estimaram o RUL de elementos eletrônicos através do *Gradient Boosting Decision Tree* (GBDT).

Assim, a estimação de RUL a partir de modelos de ML é apropriada pois além de permitir o processamento de uma ampla variedade de dados, também é capaz de aprender características intrínsecas a esses, fornecendo estimativas precisas [16]. Porém, um desafio se dá quando o número de

dados utilizados é reduzido, resultando em pouca informação disponível para o treinamento dos modelos.

Portanto, este artigo tem como objetivo analisar quatro algoritmos de aprendizagem para estimar o RUL em cinco distintos conjuntos de dados limitados. As performances dos modelos são comparadas utilizando como entrada os dados brutos, bem como, após a utilização do *Empirical Mode Decomposition* (EMD), método responsável por realizar a decomposição de uma série temporal em outras séries de frequências diferentes, permitindo uma análise mais detalhada do fenômeno [17]. Finalmente, os hiperparâmetros dos modelos serão otimizados através de uma busca em grade e validação cruzada (*GridSearchCV*, GS-CV).

O restante deste artigo está organizado da seguinte maneira: na Seção 2, será abordada uma breve fundamentação teórica sobre os quatro modelos de ML utilizados e do EMD. Na Seção 3, é apresentada a metodologia utilizada nos diferentes conjuntos de dados apresentados neste trabalho. Na Seção 4, os resultados das aplicações são discutidos e na Seção 5 são apresentadas as conclusões do artigo. É importante destacar que não são muitos os artigos de ML na área de engenharia de confiabilidade publicados em língua portuguesa, e o presente artigo também busca fornecer alcance para nativos não proficientes em outras línguas.

II. FUNDAMENTAÇÃO TEÓRICA

ML é uma área da inteligência artificial relacionada ao desenvolvimento de técnicas computacionais capazes de adquirir conhecimento de forma automática e fornecer respostas baseando-se em experiências anteriores, representada por dados [18]. Em geral, os algoritmos de ML podem ser divididos em três categorias: supervisionado, não supervisionado e semi-supervisionado. Neste artigo, modelos supervisionados serão treinados para predição do tempo falha. Especificamente, 4 modelos são analisados: *Decision Tree* (DT), *K-Nearest Neighbors* (KNN), *Multilayer Perceptron* (MLP) e *Support Vector Regression* (SVR).

DT são algoritmos de aprendizagem supervisionados e não-paramétricos usados em problemas de classificação e regressão. São baseados em uma árvore de eventos, e cada caminho a partir da raiz representa uma sequência de divisão de dados até que um resultado seja alcançado no nó folha (também chamado de nó terminal) [19]. O algoritmo começa a construir a árvore a partir do nó raiz, o qual comporta todo o espaço de entrada [20]. Este nó raiz particiona os dados de entrada em duas regiões de acordo com a variável de divisão (j) e um ponto de divisão (l), gerando dois nós filhos (Equação (1) e Equação (2)).

$$Q_t^{esq}(\gamma) = \{(x, y) | x_j \leq l_t\} \quad (1)$$

$$Q_t^{dir}(\gamma) = \{(x, y) | x_j > l_t\} \quad (2)$$

Onde Q_t é o conjunto de pontos presentes no nó t e $\gamma = (j, l_t)$ é um candidato a divisão para o nó t . A qualidade de uma divisão candidata de nó t é então calculada usando uma função de impureza (Equação (3)), Onde n_t , n_t^{esq} e n_t^{dir} são a quantidade de pontos do nó t , do nó filho esquerdo do nó t e do nó filho direito do nó t e $H(\cdot)$ é o critério a ser minimizado para determinar boas divisões, aqui representando o erro quadrático

médio. A divisão que minimiza a soma das impurezas nos dois nós filhos é escolhida.

$$I(Q_t, \gamma) = \frac{n_t^{esq}}{n_t} H(Q_t^{esq}(\gamma)) + \frac{n_t^{dir}}{n_t} H(Q_t^{dir}(\gamma)) \quad (3)$$

É então desenvolvida uma grande árvore usando um critério de parada baseado na quantidade de pontos de dados presentes nos nós de folha ($n_t < \min_{\text{samples}}$ [ver TABELA II] ou $n_t = 1$) e, em seguida, poda a árvore de forma a equilibrar o erro residual e a complexidade do modelo [20][21], uma vez que o tamanho da árvore define a complexidade do modelo. No *scikit-learn*, a poda de custo-complexidade mínima é utilizada, a qual utiliza o parâmetro de complexidade α para definir a medida de custo complexidade $C_\alpha(T)$ de uma árvore T .

KNN é um algoritmo de aprendizado no qual uma instância ainda não conhecida tem sua previsão definida de acordo com as K instâncias mais próximas, fornecidas na etapa de treinamento [22]. Para problemas de classificação, a previsão é definida de acordo com a classe com maior número de instâncias na vizinhança de K . Logo, valores diferentes de K podem gerar resultados distintos em um mesmo conjunto de dados. Já em problemas de regressão, a previsão é fundamentada na média das K instâncias mais próximas. Sendo assim, é definida para regressão da seguinte forma [20]:

$$f(x) = \frac{\sum_{x_i \in N_k(x)} (l_i \cdot Y(x_i))}{\sum_{i=1}^K l_i} \quad (4)$$

Onde x_i é a i -ésima das k instâncias mais próximas de x , k é o número de vizinhos mais próximos a serem analisados, $N_k(x)$ é a vizinhança de x definida pelas k instâncias mais próximas de x na amostra de treinamento, l_i é o peso da i -ésima instância mais próxima de x , $Y(x_i)$ é o resultado da i -ésima instância conhecida mais próxima de x , x_x é a coordenada x da instância não conhecida, $x_{i,x}$ é a coordenada x da i -ésima instância conhecida mais próxima de x e assim sucessivamente, até a coordenada n .

RNAs são algoritmos de aprendizagem projetados para simular o sistema nervoso biológico de organismos ao adquirir conhecimento através da experiência. Sua estrutura é organizada em camadas, sendo classificadas em três tipos: (i) camada de entrada, (ii) camadas intermediárias (ocultas), e (iii) camada de saída. O MLP é um tipo específico de RNA em que todos os neurônios em camadas subjacentes estão conectados. Sua estrutura pode ser representada da seguinte forma:

$$f(X) = g_s \left(\sum_{j=1}^m w_{kj} \cdot g_o \left(\sum_{i=1}^n w_{ji} \cdot X_i + b_{j_o} \right) + b_{k_s} \right) \quad (5)$$

Onde, g_s e g_o são as funções de ativação do neurônio de saída e do neurônio oculto, m e n correspondem a quantidade de neurônios na camada oculta e de entrada, w_{kj} é o peso na camada de saída conectando o j -ésimo neurônio da camada oculta e o k -ésimo neurônio da camada de saída, w_{ji} é o peso na camada oculta conectando o i -ésimo neurônio na camada de entrada e o j -ésimo neurônio na camada oculta. X_i representa as i entradas da rede neural. b_{j_o} e b_{k_s} são os vieses para o j -ésimo neurônio oculto e k -ésimo neurônio de saída.

Os pesos e os vieses são otimizados de forma a minimizar o erro entre a saída prevista e a saída verdadeira no conjunto de treinamento através do algoritmo retropropagação [23] em

conjunto com um otimizador. Aqui, três otimizadores foram testados (i.e., Adam, Limited-memory BFGS e Stochastic Gradient Descent) e quatro funções para g_o (i.e., identidade, logística, tangente hiperbólica e a *rectified linear unit* – ReLu). SVM é um dos mais populares algoritmos de ML. Pode ser utilizado em problemas de classificação e regressão. No problema de regressão, busca minimizar a distância entre o valor previsto e o hiperplano ótimo (Equação (6)) através da minimização do risco regularizado (Equação (7)), otimizando os parâmetros w e b [8].

$$f(x) = w^T x + b \quad (6)$$

$$R(C) = C \frac{1}{m} \sum_{i=1}^N V(y_i - f(x_i)) + \frac{1}{2} \|w\|^2 \quad (7)$$

Onde C é a medida de suavização da função, responsável por medir a compensação entre o risco empírico e a suavidade do modelo [24]. $V(\cdot)$, neste trabalho, é a função ε -insensível de Vapnik (Equação (8)), a qual ignora os erros dentro do tubo de raio menor que ε , isto é, a magnitude do erro não importa desde que seja menor que ε . Dessa forma, ε mede a tolerância durante a execução do processo de treinamento, tendo o valor padrão de 0,1 no *scikit-learn*.

$$V(y_i, f(x_i)) = \begin{cases} 0, & \text{se } |y_i - f(x_i)| \leq \varepsilon, \\ |y_i - f(x_i)| - \varepsilon, & \text{caso contrário} \end{cases} \quad (8)$$

Tem-se, então, o problema primal definido da seguinte forma:

$$\min_{w, \varepsilon} \left(\frac{1}{2} \|w\|^2 + C \sum_{i=1}^N \xi_i + \xi_i^* \right) \quad (9)$$

Sujeito às restrições:

$$y_i - w^T x_i - b \leq \varepsilon + \xi_i \quad (10)$$

$$w^T x_i + b - y_i \leq \varepsilon + \xi_i^* \quad (11)$$

$$\xi_i \geq 0 ; \xi_i^* \geq 0 \quad (12)$$

Onde N representa a quantidade de pontos penalizados e ξ_i e ξ_i^* representam os erros dos i -ésimos pontos penalizados acima e abaixo do tubo, respectivamente. Devido ao problema primal geralmente ser mais complexo [25], comumente é utilizado multiplicadores de Lagrange e, em seguida, a forma dual é obtida, a qual uma função kernel é aplicada, permitindo que a função $f(x)$ seja avaliada em um espaço de recursos de alta dimensão, evitando o cálculo explícito do produto interno ($w^T x$). Assim, a função de regressão é dada por:

$$f(x) = \sum_{i=1}^m (\alpha_i - \alpha_i^*) K(x_i, x) + b \quad (13)$$

Onde α_i e α_i^* são os multiplicadores de Lagrange associados às restrições do problema primal. Para mais informações, ver [26]. Neste estudo, a função de base radial (RBF) [27] é usada como a função kernel, representada por $k(x_i, x) = \exp(-\gamma \|x_i - x\|^2)$.

No contexto de engenharia de confiabilidade, os modelos de ML mencionados são vistos em aplicações da previsão de falha de diversos equipamentos e sistemas, como em baterias [28],

correia dentada [29], engrenagens [30], linhas de produção [31], motores [32] e rolamentos [33].

Adicionalmente, a fim de decompor um sinal complexo, técnicas de pré-processamento podem ser aplicadas. Neste sentido, o EMD é um processo útil para analisar séries não lineares e não estacionárias [17]. Ele decompõe a série original em uma quantidade N de outras séries, conhecidas como *Intrinsic Mode Functions* (IMFs), e um resíduo R , sendo representada Equação (14):

$$y(t) = \sum_{i=1}^N IMF_i(t) + R(t) \quad (14)$$

De modo a evitar uma maior presença de ruídos, causados por oscilações não comportadas, os IMFs são definidos de modo que os números de cruzamento de zero e extremos difiram de no máximo um, com envelopes simétricos definidos pelos máximos e mínimos locais e média zero [17]. O processo de peneiramento, usado para extrair os IMFs, é composto pela obtenção dos envelopes, a média deles e, finalmente, o IMF. Os envelopes, superior (e_{sup}) e inferior (e_{inf}), são curvas cúbicas que conectam todos os máximos locais e mínimos locais do sinal analisado via interpolação. Depois de obtidos, calcula-se a média (Equação (15)), que então é subtraída do sinal atualmente em análise (Equação (16)).

$$m_i(t) = \frac{e_{sup} - e_{inf}}{2} \quad (15)$$

$$R_i(t) = x_i(t) - m_i(t) \quad (16)$$

Em seguida, é verificado se o resultado desse processo de subtração (chamado de resíduo, $R(t)$), satisfaz as condições necessárias para ser considerado um IMF. Se não satisfizer, continua-se com o processo de peneiramento, tendo como novo sinal o resíduo gerado nesta iteração (Equação (17)); se for satisfeito, um IMF é extraído (Equação (18)) e o processo recomeça, agora com o sinal analisado sendo a subtração da série original pelo IMF obtido (Equação (19)). O processo é finalizado quando o resíduo se torna uma função monótona na qual não é mais possível obter IMFs.

$$x_{i+1}(t) = R_i(t) \quad (17)$$

$$IMF_i(t) = R_i(t) \quad (18)$$

$$x_{i+1}(t) = x_i(t) - R_i(t) \quad (19)$$

A Fig. 1 mostra um exemplo da decomposição via EMD, na qual há 3 IMFs e um resíduo. Note que a série fica mais suave nos últimos IMFs. Tal decomposição possibilita a análise de características não perceptíveis na série original, isto é, identificação e compreensão de mudanças sutis que acontecem na série. O EMD possui aplicações na área de engenharia de confiabilidade, como apresentadas em [34] e [35].

III. METODOLOGIA E APLICAÇÕES

Os algoritmos apresentados na Seção II. serão aplicados em conjuntos de dados de tamanhos variados e referentes a diferentes sistemas (como descritos na Tabela I). O primeiro dos conjuntos está relacionado a falhas de turbocompressores em um motor a diesel. Já o segundo, está relacionado com falhas em motores de automóveis (medidos em milhas), enquanto no terceiro conjunto são apresentadas informações

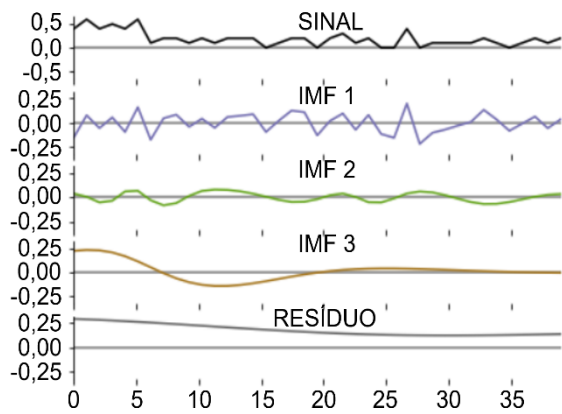


Fig. 1. Decomposição apresentada pelo EMD.

relativas a manutenções não planejadas de um motor a diesel de submarino. O quarto conjunto está relacionado com a falha de motores a jato em aviões. Por último, o quinto conjunto é alusivo a falhas em equipamentos de construção.

TABELA I
CARACTERÍSTICAS DAS CINCO BASES DE DADOS

Base	Equipamento	Tipo de Dado	Qnt. de Dados	Ref.
A	Turbocompressor	TTF (10 ³ horas)	40	[9]
B	Motor	TBF (10 ³ milhas)	100	[9]
C	Motor	TBF (dias)	70	[36]
D	Motor	TTF (horas)	66	[37]
E	Equipamentos de Construção	TBF (horas)	30	[38]

A metodologia proposta é apresentada na Fig. 2. Para a criação dos algoritmos, a biblioteca de ML *scikit-learn* [21] e o pacote ‘emd’ [39] foram utilizados em linguagem Python. Os modelos foram rodados através do Google Colab, e a máquina utilizada possui 16GB de memória RAM e processador i7 de décima geração. Todos os códigos utilizados nesse artigo estão disponíveis no Material Suplementar.

+Apesar dos avanços tecnológicos, etapas fundamentais como coleta e processamento de amostras ainda são necessárias [40]. Primeiramente é verificado qual o tipo de conjunto de dados de falha: tempo entre falhas (TBF) ou tempo de falha (TTF). No segundo caso, calcula-se o TBF uma vez que esta é a série efetivamente analisada. O TBF informa a diferença de medidas entre duas falhas consecutivas, indicando a duração que o sistema operou sem perturbações. Seu monitoramento permite identificar e compreender qualquer deterioração da confiabilidade do sistema [41], sendo uma métrica, relacionada

à disponibilidade e confiabilidade dos equipamentos. O TBF pode aumentar ou diminuir: caso aumente, tem-se uma melhoria do desempenho do sistema ao longo do tempo; caso contrário, tem-se uma piora no desempenho e, conseqüentemente, diminuição mais intensa da confiabilidade. Depois de obter o TBF, cada modelo de ML é treinado como uma série temporal, no qual o valor atual da série é usado para fazer a previsão de um período imediatamente posterior (i.e., o modelo é representado por $Y_t = f(Y_{t-1})$). Desta forma, a previsão é sempre calculada um passo-a-frente considerando apenas a amostra corrente. Em seguida, dois ramos são derivados, um com o pré-processamento via EMD e outro sem. Ao realizar o pré-processamento, obtêm-se os IMFs e o resíduo. Em seguida, é realizada a separação da série em treinamento e teste. Cerca de 80% da amostra é usada no treinamento, onde o modelo busca aprender características intrínsecas aos dados, enquanto o restante 20% dos dados são utilizados para teste. A amostra de teste é desconhecida para o modelo, tornando possível verificar a robustez e generalização do modelo, uma vez que os efeitos de *overfitting* não são sentidos aqui.

Paralelamente, na etapa de treinamento, é realizada uma otimização dos hiperparâmetros através do GS-CV, apresentados na Tabela II. GS-CV é um método do *scikit-learn* que possui um propósito duplo para ajustar modelos, sendo a combinação de hiperparâmetros o objetivo principal: primeiro, é realizado uma pesquisa de grade a uma matriz de hiperparâmetros (Tabela II); segundo, faz a validação cruzada dos modelos utilizando o método *k-fold*. Dessa forma, o GS-CV é responsável por combinar e verificar as várias combinações dos hiperparâmetro de cada modelo. Por exemplo, se para um modelo houver 3 hiperparâmetros com 5 valores cada, serão realizadas 125 combinações (5x5x5). Além disso, uma validação cruzada de 5 é feita, aumentando o número de combinações/execuções para 625. A validação cruzada *k-fold* é um processo que extrai parte da amostra de treinamento para a validação do modelo, objetivando uma maior capacidade de generalização. O processo é repetido *k* vezes para garantir uma boa divisão representação. Os hiperparâmetros são descritos abaixo, com detalhes apresentados no material suplementar:

- C: Penalidade para instâncias previstas incorretamente.
- Gamma: Largura da função do kernel (e.g., RBF).
- Max_features: Número máximo de instâncias do conjunto amostrado.
- Min_samples_split: Número mínimo de amostras necessárias para dividir um nó interno.

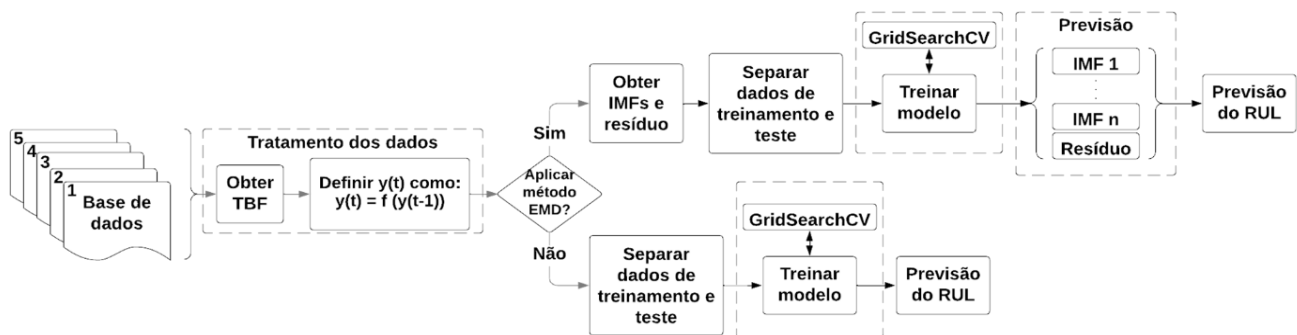


Fig. 2. Fluxograma da metodologia Modelo+EMD e Modelo puro.

- **Min_samples_leaf**: Número mínimo de amostras de cada nó folha (nó externo).
- **Activation**: Função de ativação utilizada na camada oculta da rede neural.
- **Solver**: Solucionador para otimização dos pesos.
- **Tol**: Tolerância para otimização para convergência do algoritmo.
- **N-neighbors**: Número de vizinhos mais próximos a analisados em determinada instância.
- **Weights**: Peso para a distância entre instâncias vizinhas.
- **Leaf_size**: Tamanho da folha para calcular os vizinhos mais próximos.

TABELA II
HIPER PARÂMETROS ANALISADOS

Modelo	Parâmetro	Valores
SVR	C	np.linspace(1, 1000, 25)
	gamma	np.linspace(1, 1000, 25)
	max_features	auto, sqrt, log2, None
DT	min_samples_split	range(1, 5)
	min_samples_leaf	range(1, 5)
MLP	activation	identity, logistic, tanh, relu
	solver	adam, lbfgs, sgd
	tol	1e-1, 1e-3, 1e-4, 1e-5, 1e-7
KNN	n_neighbors	range(1, 24, 1)
	weights	uniform, distance
	leaf_size	range(10, 60, 10)

Uma vez criados modelos específicos para cada um dos IMFs e resíduo, realiza-se a previsão individual de cada um deles e, em seguida, soma-se essas previsões, obtendo-se efetivamente a previsão do final do RUL, como mostrado na Fig. 3.

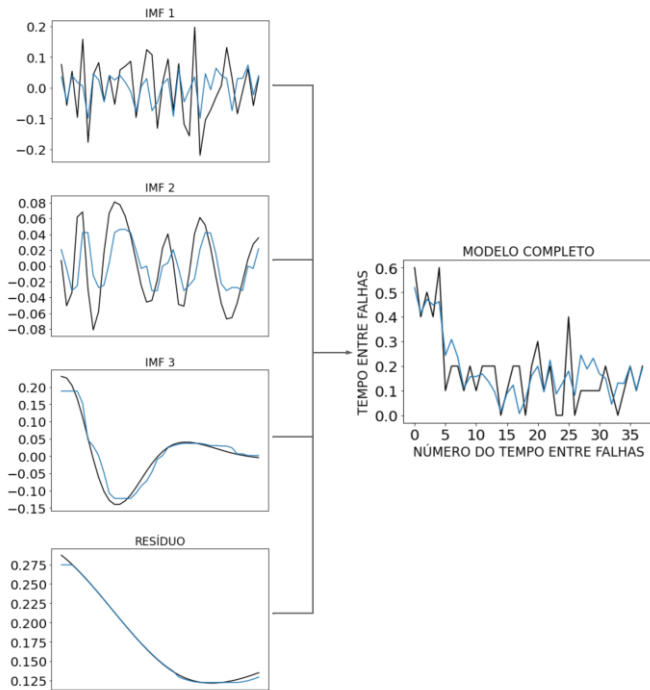


Fig. 3. Exemplo genérico de IMFs e previsões individuais.

A qualidade da previsão dos modelos depende de quão boas sejam as previsões para cada um dos IMFs. Desta forma, se o

modelo capta padrões e características específicas de cada IMF, então a previsão com EMD tenderá a acompanhar as flutuações da série original. Para o ramo sem EMD, não existem IMFs e resíduo, então após a separação dos dados o modelo é diretamente treinado e, posteriormente, obtém-se a previsão um passo à frente da série original. Todos os modelos são analisados com e sem otimização de parâmetros via GS-CV. Para analisar a performance dos modelos gerados, o erro quadrático médio normalizado (NRMSE) foi utilizada considerando apenas a amostra de teste (Equação (20)).

$$NRMSE = \frac{\sqrt{\frac{\sum_{i=1}^n (Y_{i,previsão} - Y_{i,real})^2}{n}}}{Y_{max,real} - Y_{min,real}} \quad (20)$$

onde $Y_{i,previsão}$ é o tempo previsto da falha i , $Y_{i,real}$ é o tempo real da falha i , $Y_{max,real}$ e $Y_{min,real}$ são o maior e menor tempo em que uma falha ocorreu, respectivamente, e n é o número total de dados.

IV. RESULTADOS

A. Turbocompressores em um Motor a Diesel

Neste primeiro caso foi necessário obter o TBF e, por também ser um modelo autorregressivo de primeira ordem, resultou na perda de duas amostras da série original. Assim, apenas 38 são criados, sendo os 30 primeiros para a etapa de treinamento e os últimos 8 para a etapa de teste. Em seguida, utilizou-se o EMD para fazer o pré-processamento dos dados, extraíndo-se 3 IMFs e o resíduo. Posteriormente, os quatro modelos foram treinados e seus hiperparâmetros otimizados (tal processo durou cerca de 1 minuto e 8 segundos para este caso).

Os gráficos das previsões de modelos puro e com EMD, antes da otimização de parâmetros, são apresentados apenas no Material Suplementar. Neste, é possível observar que ao pré-processamento das amostras, o modelo DT acertou os 30 primeiros tempos entre falhas, mostrando um ajuste perfeito de treinamento. Porém, tal efeito implica em uma ‘memorização’ da amostra de treinamento (i.e., *overfitting*). Nestes casos, em geral, o modelo perde poder de generalização, tendo grandes erros para as amostras de teste. Já nos modelos de MLP e KNN, a previsão com EMD aparenta melhoria em relação à sem EMD ao tentar acompanhar as flutuações da série original. A Fig. 4 apresenta os valores de NRMSE para todos os casos, onde a escala de cor de verde à vermelho representa das melhores para a piores performances.

Nos modelos com otimização dos hiperparâmetros (Fig. 5), o *overfitting* apresentado pelo DT foi evitado. Entretanto, os modelos puros de MLP e KNN tiveram uma queda na sua generalização, passando a apresentar estimativas praticamente constantes e não reconhecendo padrões (i.e., *underfitting*). Percebe-se que o modelo KNN apresentou uma melhor performance para o conjunto de amostras analisadas, com 3 dos 4 menores erros. Ainda, é possível notar que a busca de hiperparâmetros diminuiu os erros da maioria dos modelos (e.g., DT+EMD), além de evitar o *overfitting*. Aqui, o EMD foi benéfico para os modelos MLP e KNN.

		SVR		DT		MLP		KNN	
		Sem GS-CV	Com GS-CV	Sem GS-CV	Com GS-CV	Sem GS-CV	Com GS-CV	Sem GS-CV	Com GS-CV
Caso 1	Sem EMD	0,393	0,356	0,364	0,362	0,524	0,367	0,436	0,351
	Com EMD	0,691	0,646	0,486	0,38	0,318	0,36	0,298	0,307
Caso 2	Sem EMD	0,152	0,152	0,185	0,176	0,331	0,232	0,184	0,205
	Com EMD	0,151	0,151	0,215	0,175	0,162	0,156	0,172	0,177
Caso 3	Sem EMD	0,272	0,274	0,239	0,332	0,352	0,327	0,252	0,32
	Com EMD	0,263	0,304	0,619	0,398	0,225	0,27	0,300	0,282
Caso 4	Sem EMD	0,325	0,327	0,343	0,34	0,302	0,279	0,329	0,264
	Com EMD	0,299	0,286	0,368	0,37	0,288	0,288	0,322	0,27
Caso 5	Sem EMD	0,357	0,354	0,338	0,576	0,573	0,378	0,331	0,353
	Com EMD	0,373	0,297	0,486	0,415	0,483	0,489	0,441	0,472
Média		0,328	0,315	0,364	0,352	0,356	0,315	0,307	0,300
Mediana		0,312	0,301	0,354	0,366	0,325	0,308	0,311	0,295
Desvio Padrão		0,153	0,137	0,136	0,116	0,132	0,091	0,090	0,083

Fig. 4. NRMSE para todos os quatro modelos de ML aplicados aos 5 casos, com e sem EMD e GS-CV.

B. Motores de Automóvel

Uma vez que as amostras deste conjunto já são valores entre falhas, não foi necessário realizar a manipulação dos dados. Porém, devido à previsão ser um passo à frente, uma amostra ainda é perdida, restando 99 amostras ao total. Desses, os 79 primeiros para treinamento e os 20 restantes para a etapa de teste. Os passos realizados no conjunto anterior são repetidos, obtendo-se 5 IMFs e o resíduo na aplicação do EMD.

As previsões sem otimização de parâmetros são apresentadas no Material Suplementar. Novamente, o modelo DT sofre *overfitting* quando aplicado o EMD. Já o modelo puro de MLP, embora aparente estar seguindo a série real, está na verdade retardado um período e, portanto, errando bastante. Os demais modelos aparentam apresentar boa adequação aos dados, confirmada na Fig. 4. Para este conjunto, a otimização dos hiperparâmetros durou 3 minutos e 15 segundos, com previsões apresentadas na Fig. 6.

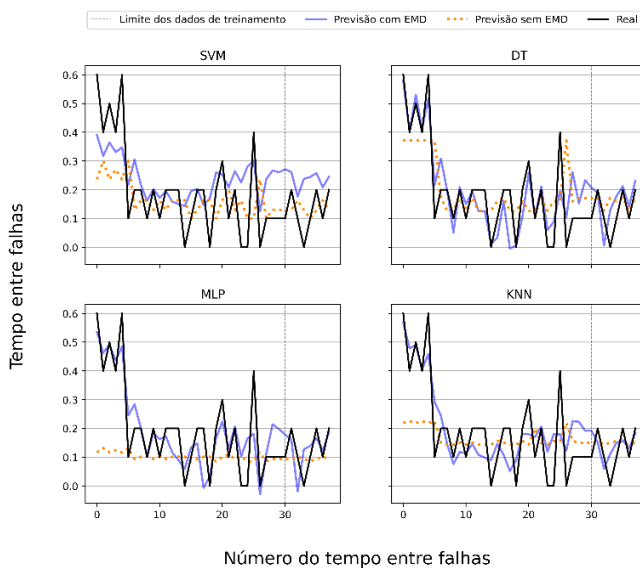


Fig. 5. Previsão de RUL para o 1º conjunto de dados com GS-CV.

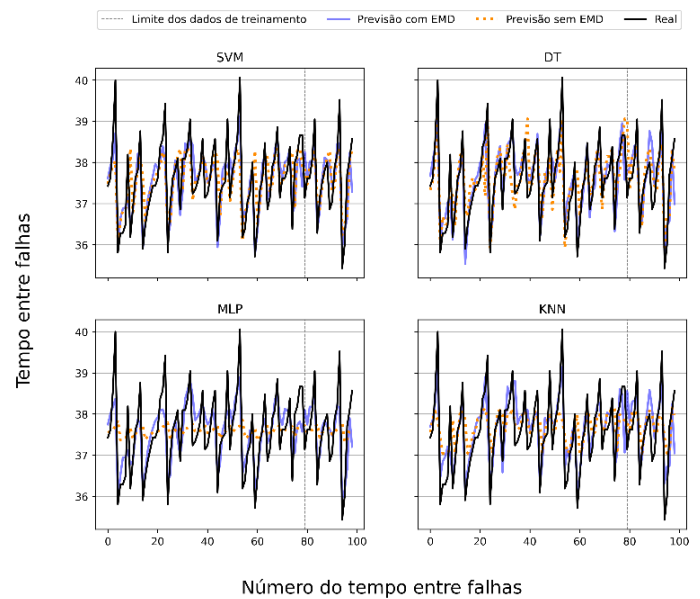


Fig. 6. Previsão de RUL para o 2º conjunto de dados com GS-CV.

Aqui, o modelo puro de MLP parou de oferecer estimativas retardadas, mas passou a apresentar estimativas praticamente constantes, entre 37 e 38. Já o SVR apresentou a melhor performance para este conjunto de amostras, tendo as melhores métricas tanto com o EMD quanto sem, fazendo pouca distinção neste processamento. Porém, nos casos de MLP e KNN, a utilização do EMD reduziu consideravelmente os erros. Em relação à otimização, o modelo MLP puro com busca sofreu *underfitting*, fazendo com que ele tivesse o maior erro em comparação aos demais. Porém, ao aplicar o método EMD o modelo apresentou previsões melhores, sempre tentando se aproximar da série original, o que resultou em uma redução de aproximadamente 32,8% do NRMSE (ver Fig. 4). De fato, a busca de hiperparâmetros causou redução nos erros dos modelos DT e MLP. No primeiro, a melhora do DT se dá por evitar o novamente *overfitting*. Já o segundo continua errando sempre, mas passou a errar na média. Logo, a diminuição dos erros não significa robustez. Após a busca, mais uma vez, o modelo SVR é o melhor para o conjunto atual.

C. Motor a Diesel de Submarino

Para este conjunto, 69 amostras foram usadas, sendo os 55 primeiros no treinamento e os 14 últimos no teste. 4 IMFs e um resíduo foram obtidos. O Material Suplementar apresenta as previsões com e sem EMD antes da otimização de parâmetros. Mais uma vez, os erros de testes são mostrados na Fig. 4.

Antes da otimização, o modelo SVR puro apresenta estimativas constantes a partir da observação 18. Após o processamento com EMD, as estimativas tentam acompanhar a série real uma vez que os IMFs buscam capturar detalhes não facilmente reconhecíveis na série original. Novamente, os modelos baseados em DT sofreram *overfitting*. Já o modelo MLP+EMD aparenta fornecer estimativas melhores que o MLP puro, com valores mais próximos da série original. Por último, o modelo KNN+EMD parece ter estimativas similares em relação ao KNN puro.

Aqui, a otimização dos hiperparâmetros durou cerca de 2 minutos e 26 segundos. A Fig. 7 mostra que a busca de hiperparâmetros também foi capaz de evitar o *overfitting* para os modelos de DT. Além disso, os resultados da Fig. 4 evidenciam que a busca reduziu os erros da maioria deles. De fato, o GS-CV foi capaz de reduzir os erros do DT+EMD em 35,7%.

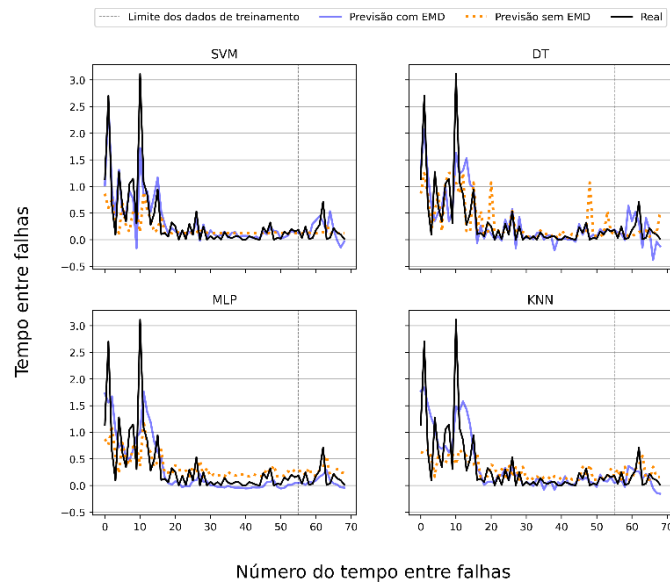


Fig. 7. Previsão de RUL para o 3º conjunto de dados com GS-CV.

D. Motores a Jato em Aviação

Manipulou-se as amostras para obter os TBF, resultando em uma série com 64 dados após a nova redução devido ao passo à frente. Destes, os 51 primeiros foram usados para o treinamento e os últimos 13 na etapa de teste. Foram obtidos 4 IMFs e um resíduo. O Material Suplementar apresenta as estimativas fornecidas pelos modelos sem otimização de parâmetros. Neste, é notado que o SVR+EMD tenta acompanhar a série original, enquanto o SVR puro sofre *underfitting*. Novamente o modelo DT+EMD sofreu *overfitting*. Finalmente, os modelos com EMD do MLP e KNN aproximam-se mais da previsão real quando comparados aos modelos puros.

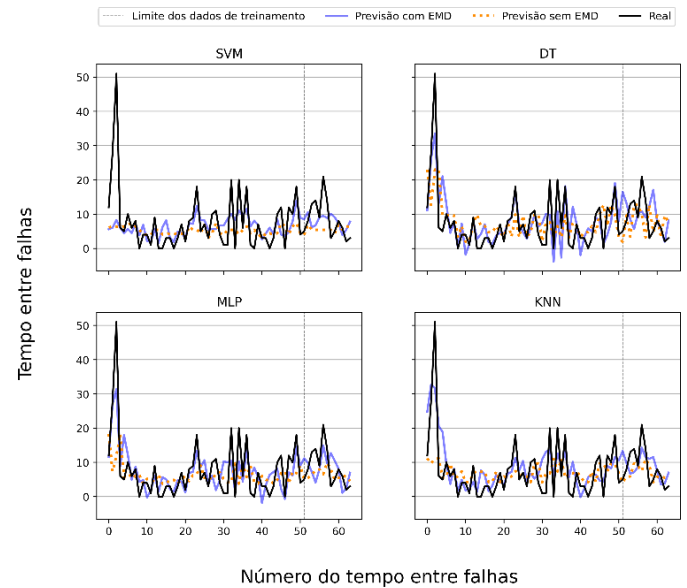


Fig. 8. Previsão de RUL para o 4º conjunto de dados com GS-CV.

Para este conjunto, a otimização dos hiperparâmetros durou cerca de 2 minutos e 20 segundos. A busca de hiperparâmetros evitou o *overfitting* do DT+EMD também neste conjunto, mas parece ainda apresentar *underfitting* para o SVR puro. Pode-se perceber que o MLP foi o que melhor se adequou às amostras deste conjunto (Fig. 8). Aqui, o EMD aprimorou o desempenho do SVR puro e com busca, os quais sofreram *underfitting*.

E. Equipamentos de Construção

Neste conjunto, 29 amostras foram utilizadas devido ao passo à frente, onde 23 são para o treinamento e 6 para o teste. Aqui, existiram 3 IMFs e um resíduo. O Material Suplementar apresenta as previsões antes da otimização de parâmetros, e nele é possível ver que o SVR e DT aparentam não ser uma boa opção, sofrendo *underfitting* e *overfitting*, respectivamente. Já os modelos MLP e KNN buscam acompanhar a série original tanto sem quanto com EMD, com previsões melhoradas com o pré-processamento das amostras, como visto na Fig. 4.

Para modelos otimizados (Fig. 9), o KNN obteve os melhores resultados sem EMD, enquanto o SVR teve destaque nos modelos com EMD.

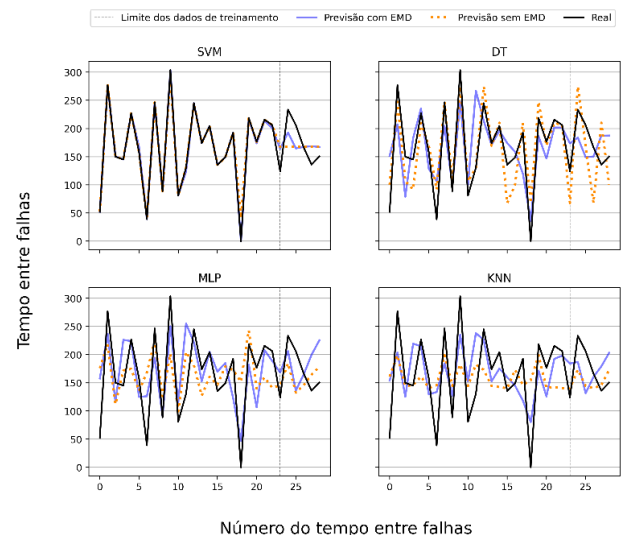


Fig. 9. Previsão de RUL para o 5º conjunto de dados com GS-CV.

V. CONCLUSÃO

O presente artigo analisou o desempenho de quatro modelos de ML em bases de dados de distintos tamanhos, buscando a previsão do RUL em pequenos conjuntos de amostras. Os modelos foram comparados em cinco bases de dados disponíveis na literatura. Ainda, o EMD foi utilizado como técnica de pré-processamento das amostras, além das buscas de hiperparâmetros dos modelos, buscando possível melhoria de performance em detrimento de um maior esforço computacional.

Devido à pequena quantidade de dados em cada conjunto, a análise de resultados das amostras de teste (20%) é complexa (e.g., modelos com *overfitting* podem ter boas performances). Aqui, aumentar a proporção do conjunto de teste provavelmente não melhoraria as performances, uma vez que a redução dos dados de treinamento poderia ao *underfitting*. Porém, para a maioria dos casos analisados, o EMD conseguiu produzir estimativas menos constantes (Figs. 5-9 e Material Suplementar). Ainda, a grande maioria dos modelos baseados em DT sofreram *overfitting*. De maneira geral os modelos apresentaram performances comparáveis, sendo identificada diferença significativa após a aplicação do teste de Kolmogorov-Smirnov (nível de significância 0.1) apenas entre o KNN sem GS-CV e o DT com GS-CV. Para estes dados, sem aplicar o EMD, os modelos SVR e KNN apresentam as melhores performances médias. Já ao considerar o pré-processamento, o MLP e KNN se destacam. De fato, para estes conjuntos de pequenas amostras, o KNN aparenta-se como promissor modelo em termos de versatilidade, apresentando menor média, mediana e desvio-padrão dos erros agregados. Presumivelmente, a busca de hiperparâmetros apresenta uma alternativa para modelos que sofrem *overfitting*, reduzindo seus erros. Dado que, nestes pequenos conjuntos de amostras, a duração máxima do GS-CV foi de menos de 3 minutos, este processo é atrativo. Sendo assim, para as bases de dados analisadas, a aplicação conjunta de busca de hiperparâmetros e EMD evitou tanto o *underfitting* como o *overfitting*. Porém, deve-se tomar cuidado ao interpretar isoladamente os menores valores de erros obtidos uma vez que, a depender do conjunto analisado, a redução nos valores médios dos erros pode ser acompanhada por previsões contantes.

Outro ponto de destaque é perceber que as performances de todos os modelos de ML são diretamente impactadas pela dimensão (quantidade) das amostras disponíveis em cada base de dados. De fato, ao analisar as médias das performances de todos os modelos (ver Fig. 4), os maiores valores são encontrados na maior base de dados (Caso 2), enquanto os menores valores são vistos nas menores bases de dados (Casos 1 e 5). Esse resultado é de certa forma esperado, uma vez que a modelos de aprendizagem a baseado em dados tornam-se mais robustos a partir da maior disponibilização de amostras [42].

É importante também mencionar que, por trabalharmos com bases de dados em unidades de medida diferentes (ver TABELA I), a padronização e normalização dos dados podem impactar as consequências dos erros de previsão para a manutenção. Por exemplo, um erro de previsão de 1 em uma na escala em dia é

bem menos impactante que o mesmo erro em uma escala de 10^3 horas.

Por fim, para trabalhos futuros, comenta-se sobre a possível inclusão de covariáveis a serem fornecidas aos modelos de ML. Esta inclusão pode ser desafiadora uma vez que os dados aqui analisados são dinâmicos e complexos. Além disso, caso haja multicolinearidade nas amostras, o modelo gerado pode se ajustar excessivamente aos dados de treinamento, perdendo generalização. Ainda, um próximo passo também será a comparação dos modelos de ML apresentados com modelos clássicos de previsão de séries temporais (e.g., ARIMA, Holt-Winter), que podem ter interpretabilidade intuitiva. Finalmente, uma análise futura também interessante é padronizar todos os bancos de dados em uma quantidade fixa de amostras (i.e., quantidade do menor banco de dados) para verificar as mudanças na performance dos modelos quando um número ainda menor de amostras está disponível.

AGRADECIMENTOS

Os autores agradecem às agências brasileira de fomento CNPq (Processo: 402761/2023-5), CAPES (Código: 001) e FACEPE (APQ-1101-3.08/21).

REFERÊNCIAS

- [1] C. B. S. Maior, M. das C. Moura, and I. D. Lins, "Particle swarm-optimized support vector machines and pre-processing techniques for remaining useful life estimation of bearings," *Eksplatacja i Niezawodnosc - Maintenance and Reliability*, vol. 21, no. 4, pp. 610–619, Sep. 2019, doi: 10.17531/ein.2019.4.10.
- [2] C. B. S. Maior *et al.*, "Seroprevalence of SARS-CoV-2 on health professionals via Bayesian estimation: a Brazilian case study before and after vaccines," *Acta Trop.*, vol. 233, p. 106551, Sep. 2022, doi: 10.1016/j.actatropica.2022.106551.
- [3] J. Lee, C. Jin, Z. Liu, and H. Davari Ardakani, "Introduction to Data-Driven Methodologies for Prognostics and Health Management," in *Probabilistic Prognostics and Health Management of Energy Systems*, Cham: Springer International Publishing, 2017, pp. 9–32. doi: 10.1007/978-3-319-55852-3_2.
- [4] C. B. S. Maior, L. M. M. Araújo, I. D. Lins, M. D. C. Moura, and E. L. Drogue, "Prognostics and Health Management of Rotating Machinery via Quantum Machine Learning," *IEEE Access*, vol. 11, pp. 25132–25151, 2023, doi: 10.1109/ACCESS.2023.3255417.
- [5] C. Okoh, R. Roy, J. Mehnen, and L. Redding, "Overview of Remaining Useful Life Prediction Techniques in Through-life Engineering Services," *Procedia CIRP*, vol. 16, pp. 158–163, 2014, doi: 10.1016/j.procir.2014.02.006.
- [6] F. C. Zegarra, J. Vargas-Machuca, and A. M. Coronado, "Tool wear and remaining useful life (RUL) prediction based on reduced feature set and Bayesian hyperparameter optimization," *Production Engineering*, Oct. 2021, doi: 10.1007/s11740-021-01086-8.
- [7] M. Tanwar and N. Raghavan, "Lubricating Oil Remaining Useful Life Prediction Using Multi-Output Gaussian Process Regression," *IEEE Access*, vol. 8, pp. 128897–128907, 2020, doi: 10.1109/ACCESS.2020.3008328.
- [8] C. B. S. Maior, M. das C. Moura, I. D. Lins, E. L. Drogue, and H. H. L. Diniz, "Remaining Useful Life Estimation by Empirical Mode Decomposition and Support Vector

- Machine,” *IEEE Latin America Transactions*, vol. 14, no. 11, pp. 4603–4610, Nov. 2016, doi: 10.1109/TLA.2016.7795836.
- [9] K. Xu, M. Xie, L. C. Tang, and S. L. Ho, “Application of neural networks in forecasting engine systems reliability,” *Appl Soft Comput*, vol. 2, no. 4, pp. 255–268, Feb. 2003, doi: 10.1016/S1568-4946(02)00059-5.
- [10] M. D. C. Moura, E. Zio, I. D. Lins, and E. Droguett, “Failure and reliability prediction by support vector machines regression of time series data,” *Reliab Eng Syst Saf*, vol. 96, no. 11, pp. 1527–1534, 2011, doi: 10.1016/j.res.2011.06.006.
- [11] S. L. Ho, M. Xie, and T. N. Goh, “A comparative study of neural network and Box-Jenkins ARIMA modeling in time series prediction,” *Comput Ind Eng*, vol. 42, no. 2–4, pp. 371–375, Apr. 2002, doi: 10.1016/S0360-8352(02)00036-0.
- [12] Z. Tian, L. Wong, and N. Safaei, “A neural network approach for remaining useful life prediction utilizing both failure and suspension histories,” *Mech Syst Signal Process*, vol. 24, no. 5, pp. 1542–1555, Jul. 2010, doi: 10.1016/j.ymsp.2009.11.005.
- [13] C. Ordóñez, F. Sánchez Lasheras, J. Roca-Pardiñas, and F. J. de C. Juez, “A hybrid ARIMA–SVM model for the study of the remaining useful life of aircraft engines,” *J Comput Appl Math*, vol. 346, pp. 184–191, Jan. 2019, doi: 10.1016/j.cam.2018.07.008.
- [14] P. J. García Nieto, E. García-Gonzalo, F. Sánchez Lasheras, and F. J. de Cos Juez, “Hybrid PSO–SVM-based method for forecasting of the remaining useful life for aircraft engines and evaluation of its reliability,” *Reliab Eng Syst Saf*, vol. 138, pp. 219–231, Jun. 2015, doi: 10.1016/j.res.2015.02.001.
- [15] L. Wang, D. Zhou, H. Zhang, W. Zhang, and J. Chen, “Application of Relative Entropy and Gradient Boosting Decision Tree to Fault Prognosis in Electronic Circuits,” *Symmetry (Basel)*, vol. 10, no. 10, p. 495, Oct. 2018, doi: 10.3390/sym10100495.
- [16] T. Wuest, D. Weimer, C. Irgens, and K.-D. Thoben, “Machine learning in manufacturing: advantages, challenges, and applications,” *Prod Manuf Res*, vol. 4, no. 1, pp. 23–45, Jan. 2016, doi: 10.1080/21693277.2016.1192517.
- [17] N. E. Huang *et al.*, “The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis,” *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1971, pp. 903–995, Mar. 1998, doi: 10.1098/rspa.1998.0193.
- [18] B. Sales da Cunha, M. das Chagas Moura, C. Souto Maior, A. Cláudia Negreiros, and I. Didier Lins, “A comparison between computer vision- and deep learning-based models for automated concrete crack detection,” *Proc Inst Mech Eng O J Risk Reliab*, p. 1748006X2211409, Dec. 2022, doi: 10.1177/1748006X221140966.
- [19] F.-J. Yang, “An Extended Idea about Decision Trees,” in *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, IEEE, Dec. 2019, pp. 349–354, doi: 10.1109/CSCI49370.2019.00068.
- [20] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. in Springer Series in Statistics. New York, NY: Springer New York, 2009. doi: 10.1007/978-0-387-84858-7.
- [21] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011, doi: 10.1007/s13398-014-0173-7.2.
- [22] S. Zhang, D. Cheng, Z. Deng, M. Zong, and X. Deng, “A novel k NN algorithm with data-driven k parameter computation,” *Pattern Recognit Lett*, vol. 109, pp. 44–54, Jul. 2018, doi: 10.1016/j.patrec.2017.09.036.
- [23] R. Rojas, “The Backpropagation Algorithm,” in *Neural Networks*, Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 149–182. doi: 10.1007/978-3-642-61068-4_7.
- [24] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2006.
- [25] A. J. Smola and B. Schölkopf, “A tutorial on support vector regression,” *Stat Comput*, vol. 14, no. 3, pp. 199–222, Aug. 2004, doi: 10.1023/B:STCO.0000035301.49549.88.
- [26] V. Kecman, “Support Vector Machines – An Introduction,” in *Support Vector Machines: Theory and Applications*, 2005, pp. 1–47. doi: 10.1007/10984697_1.
- [27] M. J. L. Orr, “Introduction to Radial Basis Function Networks,” 1996.
- [28] Y. Zhou, M. Huang, and M. Pecht, “Remaining useful life estimation of lithium-ion cells based on k-nearest neighbor regression with differential evolution optimization,” *J Clean Prod*, vol. 249, p. 119409, Mar. 2020, doi: 10.1016/j.jclepro.2019.119409.
- [29] M. Khazaei, A. Banakar, B. Ghobadian, M. A. Mirsalim, and S. Minaei, “Remaining useful life (RUL) prediction of internal combustion engine timing belt based on vibration signals and artificial neural network,” *Neural Comput Appl*, vol. 33, no. 13, pp. 7785–7801, Jul. 2021, doi: 10.1007/s00521-020-05520-3.
- [30] P. Kundu, A. K. Darpe, and M. S. Kulkarni, “An ensemble decision tree methodology for remaining useful life prediction of spur gears under natural pitting progression,” *Struct Health Monit*, vol. 19, no. 3, pp. 854–872, May 2020, doi: 10.1177/1475921719865718.
- [31] Z. Kang, C. Catal, and B. Tekinerdogan, “Remaining Useful Life (RUL) Prediction of Equipment in Production Lines Using Artificial Neural Networks,” *Sensors*, vol. 21, no. 3, p. 932, Jan. 2021, doi: 10.3390/s21030932.
- [32] Z. Chen, S. Cao, and Z. Mao, “Remaining Useful Life Estimation of Aircraft Engines Using a Modified Similarity and Supporting Vector Machine (SVM) Approach,” *Energies (Basel)*, vol. 11, no. 1, p. 28, Dec. 2017, doi: 10.3390/en11010028.
- [33] M. Yan, X. Wang, B. Wang, M. Chang, and I. Muhammad, “Bearing remaining useful life prediction using support vector machine and hybrid degradation tracking model,” *ISA Trans*, vol. 98, pp. 471–482, Mar. 2020, doi: 10.1016/j.isatra.2019.08.058.
- [34] Y. Zhou and M. Huang, “Lithium-ion batteries remaining useful life prediction based on a mixture of empirical mode decomposition and ARIMA model,” *Microelectronics Reliability*, vol. 65, pp. 265–273, Oct. 2016, doi: 10.1016/j.microrel.2016.07.151.
- [35] J. Ben Ali, N. Fnaiech, L. Saidi, B. Chebel-Morello, and F. Fnaiech, “Application of empirical mode decomposition and artificial neural network for automatic bearing fault diagnosis based on vibration signals,” *Applied Acoustics*, vol. 89, pp. 16–27, Mar. 2015, doi: 10.1016/j.apacoust.2014.08.016.
- [36] H. Ascher and H. Feingold, *Repairable Systems Reliability: Modeling, Inference, Misconceptions and Their Causes*. New York: CRC Press/Marcel Dekker, 1984.
- [37] G. R. Weckman, R. L. Shell, and J. H. Marvel, “Modeling the reliability of repairable systems in the aviation industry,” *Comput Ind Eng*, vol. 40, no. 1–2, pp. 51–63, 2001, doi: 10.1016/S0360-8352(00)00063-2.
- [38] Q. Fan and H. Fan, “Reliability Analysis and Failure Prediction of Construction Equipment with Time Series Models,” *Journal of Advanced Management Science*, vol. 3, no. 3, pp. 203–210, 2015, doi: 10.12720/joams.3.3.203-210.

- [39] A. Quinn, V. Lopes-dos-Santos, D. Dupret, A. Nobre, and M. Woolrich, "EMD: Empirical Mode Decomposition and Hilbert-Huang Spectral Analyses in Python," *J Open Source Softw*, vol. 6, no. 59, p. 2977, Mar. 2021, doi: 10.21105/joss.02977.
- [40] M. M. Aquino, C. B. S. Maior, N. A. E. Lins, C. C. B. O. Mota, P. L. A. Nascimento, and A. S. L. Gomes, "Using optical coherence tomography images to evaluate fungal growth in reline resins," *J Innov Opt Health Sci*, Jan. 2023, doi: 10.1142/S1793545822500377.
- [41] I. Alsyouf, M. Shamsuzzaman, G. Abdelrahman, and M. Al Taha, "Improving reliability of repairable systems using preventive maintenance and time-between-failures monitoring," *European J. of Industrial Engineering*, vol. 10, no. 5, p. 596, 2016, doi: 10.1504/EJIE.2016.078798.
- [42] C. B. S. Maior, M. J. das C. Moura, J. M. M. Santana, and I. D. Lins, "Real-time classification for autonomous drowsiness detection using eye aspect ratio," *Expert Syst Appl*, vol. 158, p. 113505, Nov. 2020, doi: 10.1016/j.eswa.2020.113505.



Caio Bezerra Souto Maior has a degree (2016), master (2017), doctor (2020) and post-doctor (2021) in Production Engineering from UFPE, Recife, Pernambuco, Brazil, having a sandwich degree in Industrial Engineering at the Polytechnic University of Catalonia (UPC), Terrassa, Catalonia, Spain. He is a

member of the editorial board of the journal Plos One and a regular reviewer for journals such as Reliability Engineering and Systems Safety, IEEE Transactions on Instrumentation and Measurement, IEEE Access, and Safety Science. He is currently an Adjunct Professor at UFPE (CAA Campus) and researcher at CEERMA-UFPE. His scientific interest focuses on the areas of Reliability Engineering, Machine/Deep Learning, Computer Vision, Signal Analysis and Risk Analysis.



Thaylon Gomes Silva is graduating in Production Engineering from the Federal University of Pernambuco (UFPE), Caruaru, Pernambuco, Brazil. He is currently a researcher at CEERMA-UFPE. His scientific interest focuses on the areas of Reliability Engineering, Maintenance and Machine Learning.