



Universidade do Estado do Rio de Janeiro
Centro de Tecnologia e Ciências
Faculdade de Engenharia

Marcelo Vieira Aguiar

**Aprendizado de máquina na manutenção preditiva do motor
Turbofan**

Rio de Janeiro

2023

Marcelo Vieira Aguiar

Aprendizado de máquina na manutenção preditiva do motor Turbofan

Trabalho de Conclusão de Curso apresentado, como requisito parcial para obtenção do título de Engenheiro Eletricista, a Faculdade de Engenharia, da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas e Computação.



Orientadora: Prof.^a Dra. Cristiana Barbosa Bentes

Coorientadora: Prof.^a Ms. Rafaela Correia Brum

Rio de Janeiro

2023

CATALOGAÇÃO NA FONTE
UERJ / REDE SIRIUS / BIBLIOTECA CTC/B

A283

Aguiar, Marcelo Vieira.

Aprendizado de máquina na manutenção preditiva do motor Turbofan /
Marcelo Vieira Aguiar. – Rio de Janeiro, 2023
77 f.

Orientadora: Cristiana Barbosa Bentes.

Coorientadora: Rafaela Correia Brum.

Projeto Final (Graduação) – Universidade do Estado do Rio de Janeiro,
Faculdade de Engenharia.

Bibliografia p. 74-77

1. Engenharia de sistemas e computação - Monografias. 2. Manutenção
preditiva - Monografias. 3. Máquinas e motores - Monografias. I. Bentes,
Cristiana Barbosa. II. Brum, Rafaela Correia. III. Universidade do Estado
do Rio de Janeiro, Faculdade de Engenharia. IV. Engenheiro Eletricista.

CDU 004.41

Bibliotecário: Iremar Leal da Silva - CRB7/5278

Autorizo, apenas para fins acadêmicos e científicos, a reprodução total ou parcial deste trabalho de conclusão de curso, desde que citada a fonte.

Assinatura

Data

Marcelo Vieira Aguiar

Aprendizado de máquina na manutenção preditiva do motor Turbofan

Trabalho de Conclusão de Curso apresentado, como requisito parcial para obtenção do título de Engenheiro Eletricista, a Faculdade de Engenharia, da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas e Computação.

Aprovado em 02 de junho de 2023.

Banca Examinadora:

Prof.^a Dra. Cristiana Barbosa Bentes (Orientadora)
Departamento de Engenharia de Sistemas e Computação – UERJ

Prof.^a Ms. Rafaela Correia Brum (Coorientadora)
Instituto de Computação – UFRJ

Prof.^a Dra. Karla Tereza Figueiredo Leite
Instituto de Matemática e Estatística – UERJ

Prof.^o Ms. Thiago Medeiros Carvalho
Departamento de Engenharia de Sistemas e Computação – UERJ

Rio de Janeiro
2023

DEDICATÓRIA

Aos meus pais Maria e Carlos. Ao meu irmão José Carlos e a minha cunhada Isabelle. Todos vocês contribuíram imensamente para que esta jornada fosse concluída com sucesso.

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus por iluminar o meu caminho ao longo dessa jornada.

Aos meus pais, meu irmão e minha cunhada por todo carinho, apoio, compreensão e ensinamentos que forem essenciais durante toda caminhada dentro e fora da faculdade. Vocês são minha inspiração!

Agradeço, de coração, a Larissa por toda motivação, apoio, inspiração e companhia nessa jornada.

As minhas orientadoras, professoras Cristiana e Rafaela, pelo conhecimento, orientação, incentivo, paciência e confiança em mim.

Aos professores do Departamento de Engenharia de Sistemas e Computação, pela dedicação que vocês têm a nossa universidade mesmo nos momentos difíceis que o departamento passou.

Aos membros do time da Preditiva Digital da Radix que foram de grande inspiração para o tema deste trabalho.

Aos amigos que fiz ao longo da jornada na Universidade do Estado do Rio de Janeiro.

E a todos que não foram citados, mas que contribuíram direta e indiretamente na minha formação.

“A educação é a arma mais poderosa que você pode usar para mudar o mundo.”

Nelson Mandela

RESUMO

VIEIRA AGUIAR, MVA Aprendizado de máquina na manutenção preditiva do motor Turbofan. 2023. 77 f. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2023.

Desde o início do século XXI, a manutenção preditiva (*Predictive Maintenance - PdM*) tem ganhado cada vez mais espaço na indústria, permitindo a previsão da vida útil restante (*Remaining Useful Life - RUL*) de um equipamento e evitando acidentes e prejuízos financeiros. Para isso, modelos matemáticos, incluindo modelos de aprendizado de máquina, são amplamente utilizados. Este trabalho propõe a previsão da RUL do motor Turbofan através de um modelo de aprendizado de máquina de código aberto treinado com os dados históricos desse equipamento. Para desenvolver esse modelo, foi utilizada a técnica de modelagem incremental, que envolve a divisão da modelagem em etapas menores, permitindo que cada uma seja desenvolvida, testada e aprimorada antes de prosseguir para a próxima. Foram avaliados quatro modelos diferentes (*Ridge, Lasso, Random Forest e Light Gradient Boosting Machine*) para a previsão da RUL do Turbofan, sendo que o modelo LightGBM apresentou o menor RMSE, demonstrando sua eficácia em relação aos demais. Além disso, o modelo proposto foi comparado com trabalhos relacionados e obteve uma melhoria de 32% em relação ao modelo mais complexo de redes neurais proposto por Babu *et al.* (BABU et al., 2016). Isso indica que a escolha do modelo e do pré-processamento são fundamentais para um bom desempenho, e que não existe uma solução universal para todos os dados. Além disso, um modelo preciso traz mais segurança ao setor de manutenção, evitando acidentes fatais e prejuízos financeiros.

Palavras-chave: Manutenção preditiva. Aprendizado de máquina. Motor Turbofan.
Modelagem incremental.

ABSTRACT

VIEIRA AGUIAR, MVA **Machine Learning in Predictive Maintenance of Turbofan Engine.** 2023. 77 f. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2023.

Since the beginning of the 21st century, Predictive Maintenance (PdM) has been gaining more and more ground in the industry, allowing for the prediction of the Remaining Useful Life (RUL) of equipment and avoiding accidents and financial losses. To achieve this, mathematical models, including machine learning models, are widely used. This work proposes the prediction of Turbofan motor's RUL through an open-source machine learning model trained with historical data of this equipment. To develop this model, the incremental modeling technique was used, which involves dividing the modeling into smaller stages, allowing each one to be developed, tested, and improved before moving on to the next one. Four different models (Ridge, Lasso, Random Forest, and Light Gradient Boosting Machine) were evaluated for the prediction of the Turbofan RUL, with the LightGBM model showing the lowest RMSE, demonstrating its effectiveness compared to the others. Additionally, the proposed model was compared to related works and achieved a 32% improvement compared to the more complex neural network model proposed by Babu et al. (BABU et al., 2016). This indicates that the choice of model and preprocessing are crucial for good performance, and that there is no universal solution for all data. Furthermore, an accurate model brings more safety to the maintenance sector, avoiding fatal accidents and financial losses.

Keywords: Predictive maintenance. Machine learning. Turbofan engine. Incremental modeling.

LISTA DE FIGURAS

Figura 1	- Motor Turbofan (GE90) no Museu de Arte Moderna de Nova York.	19
Figura 2	- Diagrama simplificado das peças do Turbofan.	20
Figura 3	- Detalhes das pás dos compressores e das turbinas.	21
Figura 4	- Exemplo do <i>Polynomial Features</i> em uma entrada bidimensional.	25
Figura 5	- Fluxograma do algoritmo genético.	26
Figura 6	- Ajuste dos mínimos quadrados com $X \in \mathbb{R}^2$.	28
Figura 7	- Regressão na árvore de decisão.	30
Figura 8	- Esquema simplificado para algoritmo <i>bagging</i> usado no <i>Random Forest</i> .	32
Figura 9	- Esquema simplificado para algoritmo <i>boosting</i> usado no LightGBM.	33
Figura 10	- Estrutura de dados <i>DataFrame</i> para os dados de treinamento do <i>dataset FD001</i> .	43
Figura 11	- Dados normalizados para o conjunto de treinamento do <i>dataset FD001</i> .	44

LISTA DE GRÁFICOS

Gráfico 1 - Comparação dos sensores 1 e 2 em relação aos ciclos.	41
Gráfico 2 - Comparação dos dados do sensor 2 antes e após a aplicação do filtro SG.	42
Gráfico 3 - Comparação entre as curvas <i>time</i> e RUL para o <i>dataset</i> FD001.	45
Gráfico 4 - Correlação de Pearson nos dados de treinamento do <i>dataset</i> FD001.	47
Gráfico 5 - Comparação entre os valores previstos pelo modelo LightGBM-etapa-2 e os valores reais da RUL no <i>dataset</i> FD001.	52
Gráfico 6 - Importância das variáveis para o modelo LightGBM-etapa-2 no <i>dataset</i> FD001.	53
Gráfico 7 - Comparação entre o sensor 2 do conjunto de treino e do conjunto de teste.	56
Gráfico 8 - Valor da RUL considerada relevante para o treinamento dos modelos.	62
Gráfico 9 - Comparação entre os valores previstos pelo modelo LightGBM-etapa-7 e os valores reais da RUL no <i>dataset</i> FD001.	64
Gráfico 10 - Comparação entre os valores previstos pelo modelo LightGBM-etapa-7 e os valores reais da RUL no <i>dataset</i> FD002.	66
Gráfico 11 - Comparação entre os valores previstos pelo modelo LightGBM-etapa-7 e os valores reais da RUL no <i>dataset</i> FD003	67
Gráfico 12 - Comparação entre os valores previstos pelo modelo LightGBM-etapa-7 e os valores reais da RUL no <i>dataset</i> FD004	68
Gráfico 13 - Métricas para o conjunto de teste dos modelos em cada etapa no <i>dataset</i> FD001.	69
Gráfico 14 - Comparação do RMSE entre os modelos do Mathew <i>et al.</i> (MATHEW et al., 2017) e o modelo LightGBM-etapa-7.	70
Gráfico 15 - Comparação do RMSE entre os modelos em redes neurais dos trabalhos relacionados e o modelo LightGBM-etapa-7.	71

LISTA DE TABELAS

Tabela 1	- Visão geral dos componentes da gestão de ativos.	16
Tabela 2	- Descrição dos dados dos sensores do Turbofan.	40
Tabela 3	- Resultados do experimento com PyCaret.	50
Tabela 4	- Métricas para o conjunto de teste dos experimentos da Etapa 2 sem a <i>feature time</i> .	51
Tabela 5	- Métricas para o conjunto de teste dos experimentos da Etapa 2 com a <i>feature time</i> .	51
Tabela 6	- Hiperparâmetros encontrados pelo Optuna.	54
Tabela 7	- Métricas para o conjunto de teste dos experimentos da Etapa 3 com os melhores hiperparâmetros nos modelos baseados em árvore.	54
Tabela 8	- Melhoria percentual dos modelos baseados em árvore otimizados com Optuna em relação aos modelos baseados em árvore da Etapa 2.	55
Tabela 9	- Métricas para o conjunto de teste dos experimentos da Etapa 4 com transferência dos dados de treino para os dados de teste.	57
Tabela 10	- Melhoria percentual dos modelos da Etapa 4 em relação aos modelos da Etapa 2.	57
Tabela 11	- Métricas para o conjunto de teste dos experimentos da Etapa 5 com média móvel.	58
Tabela 12	- Métricas para o conjunto de teste dos experimentos da Etapa 5 com SG.	58
Tabela 13	- Comparação percentual das métricas dos modelos da Etapa 5 com média móvel em relação aos modelos da Etapa 4	59
Tabela 14	- Comparação percentual das métricas dos modelos da Etapa 5 com filtro SG em relação aos modelos da Etapa 4.	59
Tabela 15	- Métricas para o conjunto de teste dos experimentos da Etapa 6.	60
Tabela 16	- Comparação percentual das métricas dos modelos da Etapa 6 com PF em relação aos modelos da Etapa 4.	61
Tabela 17	- Métricas para o conjunto de teste dos experimentos da Etapa 7.	62
Tabela 18	- Comparação percentual das métricas dos modelos da Etapa 7 com os dados na ROI em relação aos modelos da Etapa 4.	63
Tabela 19	- Métricas do conjunto de teste para o modelo LightGBM-etapa-7 para os <i>datasets</i> FD002 à FD004.	65

LISTA DE ABREVIATURAS E SIGLAS

RUL	<i>Remaining Useful Life</i> (Vida Útil Restante)
IoT	<i>Internet of Things</i> (Internet das Coisas)
ISO	<i>International Organization for Standardization</i> (Organização Internacional para Padronização)
NBR	Norma Brasileira
KPI	<i>Key Performance Indicator</i> (Indicador-Chave de Desempenho)
PHM	<i>Prognostics and Health Management</i> (Prognóstico e Gestão de Saúde)
PdM	<i>Predictive Maintenance</i> (Manutenção Preditiva)
NASA	<i>National Aeronautics and Space Administration</i> (Administração Nacional Aeronáutica e Espacial)
LPC	<i>Low Pressure Compressor</i> (Compressor de Baixa Pressão)
HPC	<i>High Pressure Compressor</i> (Compressor de Alta Pressão)
LPT	<i>Low Pressure Turbine</i> (Turbina de Baixa Pressão)
AI	<i>Artificial Intelligence</i> (Inteligência Artificial)
SG	SavitzkyGolay
PF	<i>Polynomial Features</i> (Recursos polinomiais)
RF	<i>Random Forest</i> (Floresta aleatória)
DT	<i>Decision Tree</i> (Árvore de Decisão)
LightGBM	<i>Light Gradient Boosting Machine</i>
GBDT	<i>Gradient Boosting Decision Tree</i>
GOSS	<i>Gradient-based One-Side Sampling</i>
EFB	<i>Exclusive Feature Bundling</i>
AutoML	<i>Automated Machine Learning</i> (Aprendizado de Máquina Automatizado)
MAE	<i>Mean Absolute Error</i> (Erro Médio Absoluto)
RMSE	<i>Root Mean Squared Error</i> (Raiz do Erro Quadrático Médio)
ROI	<i>Region of Interest</i> (Região de Interesse)
LSTM	<i>Long Short Term Memory</i> (Mémória de longo e curto prazo)
DNN	<i>Deep Neural Network</i> (Rede Neural Profunda)
DCNN	<i>Deep Convolutional Neural Network</i> (Rede Neural Convolucional Profunda)
CNN	<i>Convolutional Neural Network</i> (Rede Neural Convolucional)

SUMÁRIO

	INTRODUÇÃO	14
1	GESTÃO DE ATIVOS E MANUTENÇÃO	16
1.1	Manutenção corretiva	17
1.2	Manutenção preventiva	17
1.3	Manutenção preditiva	17
1.3.1	<u>Ferramentas para aplicação da manutenção preditiva</u>	18
1.4	Motor Turbofan	18
1.4.1	<u>Componentes e suas funções</u>	19
1.4.2	<u>Vida útil restante</u>	20
2	APRENDIZADO DE MÁQUINA	22
2.1	Transformação dos dados	23
2.1.1	<u>Normalização padrão</u>	23
2.1.2	<u>Suavização dos dados</u>	24
2.1.3	<u>Recursos polinomiais</u>	25
2.1.4	<u>Algoritmo genético</u>	26
2.2	Modelos de aprendizado de máquina	27
2.2.1	<u>Régressão Ridge</u>	27
2.2.2	<u>Régressão Lasso</u>	29
2.2.3	<u>Régressão Random Forest</u>	29
2.2.4	<u>Régressão Light Gradient Boosting Machine</u>	31
2.3	Métricas de avaliação	33
2.4	Busca de hiperparâmetros	34
2.5	Aprendizado de máquina automatizado	35
3	TRABALHOS RELACIONADOS	36
4	PREPARAÇÃO DOS DADOS	38
4.1	Extração dos dados brutos	38
4.2	Análise exploratória	39
4.3	Pré-processamento	43
5	MODELAGEM INCREMENTAL	48
5.1	Experimentos com <i>dataset FD001</i>	49
5.1.1	<u>Etapa 1: Seleção do melhor modelo com AutoML</u>	49
5.1.2	<u>Etapa 2: Análise dos modelos com parâmetros padrões</u>	50
5.1.3	<u>Etapa 3: Melhoria dos modelos baseados em árvore</u>	53
5.1.4	<u>Etapa 4: Transferência dos dados de treino para os dados de teste</u>	55
5.1.5	<u>Etapa 5: Atenuação dos ruídos nos dados de entrada</u>	57
5.1.6	<u>Etapa 6: Adição de features polinomiais</u>	59

5.1.7	<u>Etapa 7: Treinamento e teste na região de interesse</u>	61
5.2	Experimento com outros datasets	63
5.3	Discussão dos resultados	65
	CONCLUSÕES	72
	REFERÊNCIAS	74

INTRODUÇÃO

O gerenciamento da manutenção e o prognóstico de um equipamento são muito importantes em diversas áreas, como aeroespacial, petroquímica, elétrica e alimentícia. A manutenção é uma atividade crítica que evita a perda ou a troca completa do equipamento, evitando um desastre e prejuízos para as empresas. Por esse motivo, diferentes estratégias de manutenção têm sido aplicadas nesses campos. Essas estratégias podem ser classificadas em três tipos: manutenção corretiva, manutenção preventiva e manutenção preditiva (YUREK; BIRANT, 2019).

A manutenção corretiva consiste nas operações de manutenção do equipamento após a ocorrência de falhas. Já a manutenção preventiva busca reduzir as falhas do equipamento a partir de reparos com intervalos pré-determinados. A manutenção preditiva (*Predictive Maintenance - PdM*), diferentemente da corretiva e da preventiva, tem como objetivo prever a vida útil restante (*Remaining Useful Life - RUL*) do equipamento. A partir dessa informação, é possível determinar, de forma mais assertiva, o período de manutenção antes que o equipamento falhe. A RUL pode ser prevista com base nos dados históricos (parâmetros de condição e desempenho). Dessa forma, são evitadas as falhas inesperadas e o excesso de manutenções pré-determinadas que geram custos extras. Para determinar a RUL, é geralmente necessário realizar a digitalização da empresa, a fim de disponibilizar os dados para o treinamento do modelo matemático que prevê a vida útil restante. Essa digitalização envolve a instalação de sensores para coleta de dados e a utilização de um banco de dados, normalmente na nuvem, para armazenar essas informações. É importante destacar que todo esse processo tem um custo associado. No entanto, de acordo com Kardec *et al.* (KARDEC et al., 2001), a relação custo/benefício da PdM para monitoramento online do equipamento é de 1/5, o que evidencia os potenciais benefícios em relação aos custos envolvidos.

Nos últimos anos, a transformação digital, a indústria 4.0 e a Internet das Coisas (Internet of Things - IoT) impulsionaram o aumento da utilização da PdM no setor industrial. Como resultado, muitas empresas têm instalado sensores em seus equipamentos, permitindo a coleta de dados. Esses dados tem informações úteis para determinar a RUL dos equipamentos. A partir desse ponto, a previsão da próxima manutenção se torna um desafio que é abordado pela Ciência de Dados (MATHEW et al., 2017).

Existem diversas ferramentas pagas disponíveis para a realização da PdM de equipamentos industriais (MUSSO, 2022). No entanto, neste trabalho foi desenvolvida uma ferramenta gratuita e de código aberto usando modelos de aprendizado de máquina (do inglês, *machine learning*) para a previsão da RUL de um motor Turbofan (HALL, 2021). Para isso, foi utilizado um conjunto de dados público disponibilizado pela NASA (National Aeronautics and Space Administration) (TEUBERT, 2007).

O objetivo deste trabalho é prever RUL de motores Turbofan, para isso foram investigados modelos baseados em aprendizado que visam realizar a previsão da RUL com o menor erro possível. Para alcançar esse objetivo, foi utilizada a técnica de modelagem incremental, que consiste em aprimorar o modelo em etapas, utilizando diferentes recursos no pré-processamento dos dados. Além disso, foram avaliados quatro algoritmos distintos (*Ridge*, *Lasso*, *Random Forest* e *Light Gradient Boosting Machine* [LightGBM]) para a previsão da RUL do Turbofan. A escolha desses algoritmos baseou-se em trabalhos relacionados e na primeira etapa da modelagem incremental. Entre os algoritmos testados, o LightGBM apresentou o menor RMSE (*Root Mean Square Error*). Por esse motivo, ele foi comparado com trabalhos relacionados e demonstrou uma melhoria de 32% em relação ao modelo mais complexo de redes neurais proposto por Babu *et al.* (BABU *et al.*, 2016) na previsão do conjunto de dados FD002.

O modelo desenvolvido poderá permitir aos engenheiros de manutenção identificar o momento em que o motor falhará. Isso possibilitará que os profissionais planejem a manutenção com antecedência, evitando paradas não programadas e reduzindo os custos com reparos.

Estrutura do trabalho

Este trabalho é composto por cinco capítulos organizados da seguinte forma:

O Capítulo 1 apresenta o setor de manutenção industrial, detalhando as diferentes práticas de manutenção e descrevendo toda a estrutura do motor Turbofan, incluindo o problema de sua degradação.

No Capítulo 2, são abordadas as técnicas de pré-processamento de dados e os modelos de aprendizado de máquina. Além disso, são apresentadas métricas de avaliação e técnicas de otimização dos modelos.

O Capítulo 3 apresenta os trabalhos relacionados à previsão da RUL do motor Turbofan por meio de modelos de aprendizado de máquina, desenvolvidos e publicados pela comunidade acadêmica.

O Capítulo 4 apresenta a base de dados do motor e as primeiras análises realizadas.

No Capítulo 5, são apresentados os experimentos realizados, bem como as métricas e a comparação do modelo desenvolvido com outros modelos da literatura.

Por fim, a conclusão apresenta uma argumentação conclusiva com base nos resultados obtidos e aponta para novos estudos sobre a modelagem da RUL para o motor Turbofan.

1 GESTÃO DE ATIVOS E MANUTENÇÃO

De acordo com a norma ISO (*International Organization for Standardization*) 55000 (ISO 55000, 2014), um ativo é um item que possui valor para uma organização, e a gestão de ativos é uma atividade coordenada para extrair valor desses itens. A gestão de ativos em uma indústria considera todo o ciclo de vida dos equipamentos, desde a elaboração da compra até o seu descarte (ZAMPOLLI et al., 2019). Isso envolve os custos e impactos do investimento antes da compra, bem como a consideração das oportunidades, riscos, custos de manutenção, ciclos de operação e depreciação do equipamento após a compra. Além disso, a gestão de ativos também contempla os custos de reforma, substituição e descarte do equipamento ao final de sua vida útil.

A manutenção é um conjunto de atividades realizadas para garantir que um determinado ativo (equipamento, máquina ou sistema) esteja em pleno funcionamento. Segundo Blanchard *et al.* (BLANCHARD et al., 1995), a manutenção é uma combinação de todas as ações necessárias para manter ou restaurar um item para seu estado efetivo de operação. Já a gestão da manutenção está incorporada à gestão de ativos, sendo uma atividade de gerenciamento que tem como objetivo reduzir reparos e substituições em máquinas. Em outras palavras, a gestão da manutenção busca prevenir manutenções desnecessárias. Além disso, ela assegura a disponibilidade e confiabilidade das máquinas, aplicando as melhores técnicas ao menor custo (MUSSO, 2022). Dessa forma, otimiza-se a vida útil do equipamento e evita-se falhas inesperadas. A relação entre a gestão de ativos e a gestão da manutenção é ilustrada na Tabela 1. Na tabela, é possível observar que a gestão de ativos abrange todas as fases, desde o planejamento até o descarte. Além disso, a gestão da manutenção está integrada à gestão de ativos, englobando as atividades de manutenção realizadas nos ativos.

Tabela 1 - Visão geral dos componentes da gestão de ativos.

Gestão de ativos				
Planejamento	Tomada de decisão	Ciclo de vida do ativo		Descarte
		ENGENHARIA	MANUTENÇÃO	
		Compra/Fabricação	Operação	
		Instalação/Montagem	Manutenção do ativo	
		Treinamento	Reparos/Substituições	
			Gestão da Manutenção	

Até poucos anos atrás, as empresas não atribuíam grande importância ao setor de manutenção. Era comum considerá-lo como uma área responsável por gastos excessivos e prejuízos. Porém, com a Indústria 4.0 e a IoT, a manutenção se intensificou como uma

ferramenta estratégica para redução de custos, aumento da produtividade e melhoria da segurança das plantas industriais (MUSSO, 2022). Existem diferentes estratégias de manutenção que podem ser usadas para alcançar bons resultados, sendo elas: manutenção corretiva, manutenção preventiva e a manutenção preditiva.

1.1 Manutenção corretiva

O conceito de manutenção corretiva surgiu na Primeira Guerra Mundial, por volta de 1914 (BARBOSA, 2018). Essa época ficou marcada pela primeira geração da manutenção, sendo caracterizada por ser improvisada e não organizada, ou seja, os serviços de limpeza, lubrificação e reparo eram feitos após a falha do equipamento. Segundo a NBR (Norma Brasileira) 5462 (NBR 5462, 1994), esse tipo de manutenção é efetuada após a ocorrência de uma pane destinada a recolocar um ativo em condições de executar uma função requerida. Isso significa que o desempenho abaixo do esperado ou a ocorrência de falha em um ativo podem ocasionar esse tipo de manutenção (MARTINS, 2022).

1.2 Manutenção preventiva

Com passar dos anos, surgiu a necessidade de manter os equipamentos operando sem falhas inesperadas. Uma planta que não está operando devido a um equipamento quebrado resulta em um prejuízo financeiro significativo. Esse prejuízo inclui o custo da manutenção para consertar o equipamento e o custo do tempo que ele deixa de produzir durante o período de reparo. Entre 1930 e 1970, as indústrias estavam bastante dependentes das máquinas, e isso levou a ideia de que as falhas nos equipamentos poderiam ser evitadas, o que sucedeu no conceito de manutenção preventiva (KARDEC et al., 2001). Dessa forma, a segunda geração da manutenção é caracterizada por reduzir a ocorrência de falhas por meio de planejamento de inspeções e reparos com intervalos predeterminados ou de acordo com critérios prescritos (NBR 5462, 1994). Nessa abordagem, as falhas são geralmente evitadas, mas, por outro lado, a quantidade de manutenções pode resultar em um aumento do custo final. Essas ações podem levar ao uso ineficiente de recursos e ao aumento dos custos, pois não otimizam a vida útil do equipamento (MARTINS, 2022).

1.3 Manutenção preditiva

A partir de 1975, percebeu-se um aumento significativo nos custos de manutenção em relação aos custos operacionais. Diante dessa realidade, a indústria passou a buscar

maneiras de estender a vida útil dos equipamentos (KARDEC et al., 2001). Paralelamente, houve uma redução nos preços dos computadores naquela época, o que viabilizou o desenvolvimento da manutenção preditiva como a terceira geração (BARBOSA, 2018). A PdM busca aplicar técnicas de análise avançada, como a previsão da vida útil restante (RUL) e o cálculo do indicador-chave de desempenho (Key Performance Indicator - KPI) do equipamento, com o objetivo de minimizar a necessidade de manutenção corretiva e preventiva (NBR 5462, 1994). Dessa forma, a PdM utiliza os parâmetros de condição e desempenho do ativo para prever a sua RUL, possibilitando maximizá-la. Com essa abordagem, é possível programar a manutenção do ativo com antecedência, uma vez que fornece informações precisas sobre o fim de sua vida útil. Isso permite um planejamento mais eficiente das atividades de manutenção e uma melhor utilização dos recursos disponíveis.

Na literatura, o problema de previsão da RUL e outras condições do sistema é conhecido como Prognóstico e Gestão de Saúde (*Prognostics and Health Management - PHM*) (SAXENA et al., 2008). Nessa área, são usados diferentes algoritmos de aprendizado de máquina para prever a RUL. Também foram usados modelos físicos, porém eles não apresentaram resultados satisfatórios (PENG et al., 2021). Um dos desafios para treinar os modelos de aprendizado de máquina é a falta de dados, já que existem poucos conjuntos de dados com a operação do equipamento até a falha. Por esse motivo, este trabalho usará os dados do motor Turbofan para determinar a RUL.

1.3.1 Ferramentas para aplicação da manutenção preditiva

Para uma empresa aplicar os princípios da PdM é necessário realizar um investimento em infraestrutura, de modo a permitir a coleta dos dados do ativo e a construção do modelo matemático para prever a RUL.

Existem diversas soluções no mercado que disponibilizam ferramentas para aplicação da PdM. Algumas delas são: Aveva (AVEVA, 2023), General Eletrics (General Eletrics, 2023) e Tractian (Tractian, 2021). Todas essas ferramentas são pagas e não informam como foram desenvolvidas. Por essa razão, este trabalho usará código aberto para desenvolver o modelo de aprendizado de máquina para prever a RUL do motor Turbofan.

1.4 Motor Turbofan

O motor Turbofan é uma variação moderna do motor básico de turbina a gás. Esse motor é utilizado como sistema de propulsão em aeronaves, sendo responsável por gerar o empuxo necessário para movimentar a aeronave no ar. (HALL, 2021). O fenômeno que

gera o empuxo segue o princípio da Terceira Lei de Newton (Lei da Ação e Reação), ou seja, é deslocada uma grande massa de ar para trás em alta velocidade que, consequentemente, gera uma reação (empuxo) que desloca o avião para frente (SILVA et al., 2017). Por esse motivo, o Turbofan também é chamado de motor a reação. Ele é altamente complexo, e é considerado o coração do avião, já que é responsável pelo deslocamento, pressurização e a produção de energia elétrica. A Figura 1 mostra um exemplo desse motor.

Figura 1 - Motor Turbofan (GE90) no Museu de Arte Moderna de Nova York.



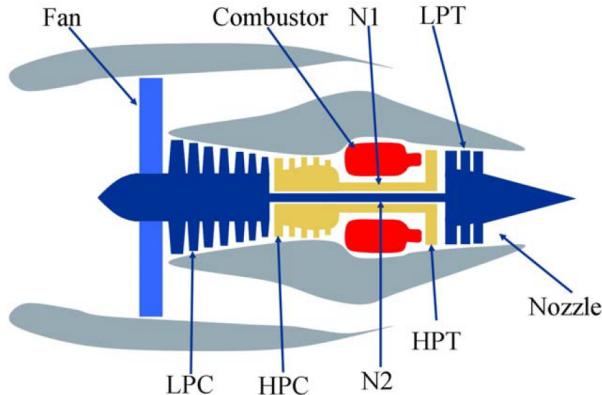
Fonte: (GE do Brasil, 2016).

1.4.1 Componentes e suas funções

Geralmente, o Turbofan é dividido em três módulos. O primeiro módulo é composto pela entrada de ar (*fan*) e o compressor de baixa pressão (*Low Pressure Compressor* - LPC). O segundo módulo é composto pelo compressor de alta pressão (*High Pressure Compressor* - HPC), o combustor e a turbina de alta pressão (*High Pressure Turbine* - HPT). Já o terceiro módulo é composto pela turbina de baixa pressão (*Low Pressure Turbine* - LPT) e a descarga (*Nozzle*). Por fim, o eixo N1 liga o HPC à HPT, e o eixo N2 interliga o LPC à LPT. A Figura 2 mostra de forma simplificada a localização de cada dispositivo dos módulos.

Durante a operação do motor, o ar é direcionado para a *fan* e uma parte dele segue um caminho contornando o núcleo do motor, conhecido como *bypass*, evitando assim a interação direta com os componentes internos.. Por esse motivo, a *fan* é responsável por produzir 80% do empuxo do motor (SILVA et al., 2017). Em seguida, o ar restante flui em direção ao núcleo e encontra primeiro o LPC. Os compressores (LPC e HPC) são

Figura 2 - Diagrama simplificado das peças do Turbofan.



Fonte: (SAXENA et al., 2008).

responsáveis por comprimir o ar e adicionar energia na forma de pressão e calor.

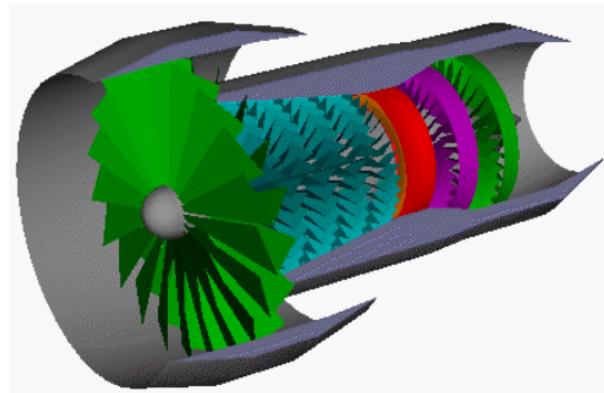
Após sair do HPC, o ar é direcionado para o combustor, onde, em alta pressão e temperatura, ele é misturado com combustível para ser queimado na câmara de combustão (THAKKAR et al., 2022). A reação química na câmara ocorre a uma temperatura de aproximadamente 1000°C, produzindo gases de alta velocidade que, em seguida, são direcionados para as turbinas. As turbinas (HPT e LPT) convertem a energia do gás, que está em alta pressão e temperatura, em energia mecânica. Essa energia impulsiona a rotação tanto dos compressores quanto da *fan*, sendo transmitida por meio dos eixos N1 e N2. Por fim, o formato ideal da descarga auxilia na regulação da pressão do motor, redução de ruído e, consequentemente, na produção de mais empuxo. A Figura 3 mostra que os compressores e as turbinas são formadas por pás giratórias que possuem um formato aerodinâmico para otimizar as suas funções dentro do motor.

1.4.2 Vida útil restante

Conforme relatado por Thakkar *et al.* (THAKKAR et al., 2022), o motor Turbofan é exposto a diversas condições ambientais e operacionais ao longo de sua vida útil, tais como corrosão, desgaste, flambagem e erosão. Esses fatores podem comprometer o motor, aumentando o desgaste, o risco à falhas e, consequentemente, colocando vidas em perigo. Por isso é fundamental monitorar e avaliar as métricas de integridade da saúde do motor para otimizar a vida útil, o desempenho e a sua confiabilidade. No entanto, devido à complexidade desses motores, falta instrumentos para avaliar de forma assertiva os parâmetros de saúde, ou seja, especificar a RUL.

Segundo Peng *et al.* (PENG et al., 2021), 60% do total das falhas de uma aeronave

Figura 3 - Detalhes das pás dos compressores e das turbinas.



Fonte: (HALL, 2021).

estão relacionadas ao motor Turbofan. Por isso, é de suma importância determinar de forma precisa a RUL do motor para fazer a manutenção e evitar qualquer tipo de tragédia. A RUL é medida em ciclos, que, por sua vez, podem ser interpretados como a quantidade de voos realizados pela aeronave. Logo, prever a RUL do Turbofan é determinar a quantidade de voos que ele pode operar.

2 APRENDIZADO DE MÁQUINA

O aprendizado de máquina é um subcampo da Inteligência Artificial (Artificial Intelligence - AI) que se refere à capacidade de uma máquina aprender com dados e realizar tarefas sem a necessidade de programação explícita para tal finalidade (BROWN, 2021).

Atualmente, o aprendizado de máquina é amplamente utilizado em diversas aplicações, tais como reconhecimento facial, detecção de fraudes bancárias, assistentes virtuais, diagnóstico de câncer, previsão do tempo, identificação de perfis em jogos e *streaming* de vídeo, entre outras. (FACELI et al., 2011). Uma pesquisa da IBM mostra que 41% das empresas brasileiras utilizam alguma forma de inteligência artificial em seu dia a dia (IBM, 2022). O relatório ainda afirma que 66% das empresas participantes indicaram que têm planos de utilizar AI para alguma aplicação envolvendo negócios.

Existem diferentes tipos de aprendizados de máquina que são utilizados conforme a sua aplicação. Alguns deles são aprendizado supervisionado e o aprendizado não supervisionado. No aprendizado supervisionado os dados de treinamento são rotulados, ou seja, para cada entrada disponível, existe um rótulo associado a este (GÉRON, 2019). Como, por exemplo, treinar um algoritmo com fotos, fornecendo a informação de quais fotos são de cachorros e quais são de gatos. Após o treino, o algoritmo pode ser capaz de dizer qual é o animal que está na foto. No aprendizado não supervisionado, os dados de treinamento não são rotulados. Em outras palavras, isso significa que o algoritmo deverá encontrar padrões e tendências nos dados que são fornecidos. No caso do exemplo anterior, o modelo é treinado apenas fornecendo as fotos sem a informação de quais fotos são de cachorros e de gatos. O algoritmo poderá, por conta própria, identificar qual animal está presente nas fotos, agrupando-as com base em características comuns, como tamanho do animal, formato da cabeça, entre outras. No entanto, é importante ressaltar que, no contexto deste trabalho, foram utilizados dados estruturados, ou seja, dados organizados em formatos específicos que facilitam a análise e aplicação de algoritmos de aprendizado de máquina.

Para cada tipo de problema existem diferentes algoritmos usados para implementar os modelos de aprendizado de máquina. Os algoritmos supervisionados podem ser divididos entre algoritmos de regressão e algoritmos de classificação. Os algoritmos de regressão são utilizados para prever um resultado quantitativo (valores numéricos) com base em variáveis independentes. Por outro lado, os algoritmos de classificação são empregados para prever variáveis categóricas, ou seja, aquelas que possuem categorias distintas. Por exemplo, eles podem ser usados para prever respostas binárias como “sim/não” ou para atribuir categorias como “gato/cachorro/passáro”. (BROWN, 2021).

Nas próximas seções, serão abordadas as transformações necessárias nos dados,

seguidas pela apresentação dos modelos de aprendizado de máquina para regressão utilizados neste trabalho. É importante destacar que essas transformações são essenciais para o treinamento dos modelos, pois sem elas o desempenho pode ser comprometido.

2.1 Transformação dos dados

A transformação de dados é um processo de modificação dos dados para uma forma adequada e utilizável em um modelo de aprendizado de máquina. O conjunto de dados para treinar um modelo possui diferentes características, sendo necessário aplicar uma estratégia diferente para cada uma delas, como, por exemplo:

- Atributos categóricos: São valores representados por caracteres, sendo necessário convertê-los para valores numéricos, uma vez que os modelos que utilizam números para fazer o ajuste nos dados de treinamento.
- Dados nulos: É necessário tratar esses dados preenchendo-os com a média, removendo-os ou outra mais adequada aos dados, a fim de evitar potenciais distorções no modelo.
- Dados duplicados: Se esses dados não trouxerem informações relevantes para o modelo, devem ser removidos para evitar que afetem negativamente a eficácia da análise.

As próximas seções apresentam de forma detalhada outras transformações que podem ser utilizadas, as quais foram efetivamente aplicadas neste trabalho.

2.1.1 Normalização padrão

É comum que os dados numéricos apresentem uma magnitude diferente no mesmo conjunto de dados. Como, por exemplo, uma pressão com valor igual a 2 atm e uma vazão com valor igual a 2000 kg/h. A distância numérica entre esses dados pode interferir no treinamento de um modelo, passando para ele a informação falsa de que a vazão é mais importante por ter o maior valor. Para evitar que um atributo predomine sobre o outro, é usado uma técnica de normalização de dados. A normalização padrão é um exemplo, onde é recomendável utilizá-la quando a distribuição dos dados é gaussiana (FACELI et al., 2011). Outro exemplo é a normalização Min-Max

O objetivo da normalização padrão é deixar os dados na mesma escala, colocando a média igual a zero e o desvio padrão igual um, conforme a equação (1).

$$z = \frac{x - \mu}{\sigma} \quad (1)$$

onde, z é o valor normalizado, x é o valor a ser normalizado, μ é a média e σ é o desvio padrão do conjunto de dados a ser normalizado.

2.1.2 Suavização dos dados

A suavização (filtragem) dos dados consiste em reduzir o ruído presente nos dados. A suavização pode melhorar a performance do modelo, mas também pode piorá-la pois o ruído pode conter informação importante. Existem algumas técnicas de suavização comumente utilizadas, entre elas estão a média móvel e o filtro de Savitzky-Golay.

1. **Média móvel:** Esse tipo de suavização é dada pela média de n períodos de entrada (janela) para produzir uma saída, de acordo com a equação (2). Ela reduz o ruído branco enquanto mantém uma tendência bem definida (NISHIDA, 2017).

$$y_i = \frac{1}{n} \cdot \sum_{j=1}^n x_{i+j} \quad (2)$$

onde, n é o tamanho da janela, y_i é a média móvel na posição i e x_{i+j} é a entrada de cada elemento da janela.

2. **Filtro de Savitzky-Golay:** O filtro Savitzky-Golay (SG) pode ser visto como uma generalização da média móvel. Ele reduz o ruído aleatório e mantém a forma da curva original. Os coeficientes desse filtro são calculados a partir de um ajuste linear não ponderado do método dos mínimos quadrados, utilizando um polinômio de ordem k para se ajustar aos dados (NISHIDA, 2017). A equação (3) mostra uma forma de calculá-lo.

$$y(t) = \sum_{i=0}^n c_i \cdot x(t - i) \quad (3)$$

Nesta equação, $y(t)$ representa o valor suavizado no instante de tempo t , $x(t - i)$ representa os valores de entrada em instantes de tempo anteriores, c_i são os coeficientes do filtro de Savitzky-Golay e n é a ordem do filtro. Os coeficientes de SG podem ser calculados pela equação (4).

$$c_i = (X^T X)^{-1} X^T y \quad (4)$$

onde X é uma matriz que contém os termos de Vandermonde dos pontos de entrada $x(t - i)$, y é um vetor que contém os valores correspondentes de saída suavizados e T indica a transposição da matriz.

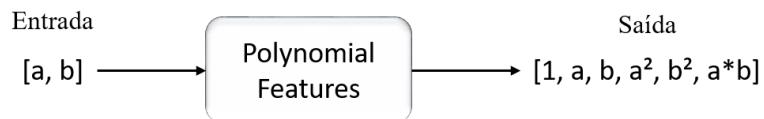
Os termos de Vandermonde são calculados a partir dos pontos de entrada $x(t - i)$ e a ordem do filtro n , conforme a equação (5).

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^n \\ 1 & x_2 & x_2^2 & \dots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \dots & x_m^n \end{bmatrix} \quad (5)$$

2.1.3 Recursos polinomiais

A técnica de transformação de dados conhecida como recursos polinomiais (*Polynomial Features* - PF) cria novas *features* por meio da combinação polinomial das *features* existentes, como exemplificado na Figura 4.

Figura 4 - Exemplo do *Polynomial Features* em uma entrada bidimensional.



O objetivo dessa transformação é melhorar o desempenho do modelo, principalmente os lineares. As *features* criadas pela PF permite que os modelos lineares reconheçam padrões não lineares nos dados. A desvantagem é que a dimensão do problema (quantidade de entradas) aumenta significativamente. Dessa forma, a regressão se torna mais complexa sendo necessário usar um algoritmo de seleção de variáveis, como um algoritmo genético. Neste trabalho, devido à utilização do PF, foi necessário empregar o algoritmo genético para selecionar as melhores variáveis.

2.1.4 Algoritmo genético

O algoritmo genético é um método dos algoritmos evolutivos que busca imitar os mecanismos da evolução biológica para resolver problemas de busca e seleção. Sua finalidade é encontrar um ponto ótimo que atenda às exigências do problema de forma eficiente (PINTO, 2022). Esse algoritmo é capaz de selecionar as melhores variáveis para o problema, reduzindo a dimensão do modelo a ser treinado. No entanto, essa vantagem vem acompanhada de um maior custo computacional, o que significa que o tempo de execução pode ser maior.

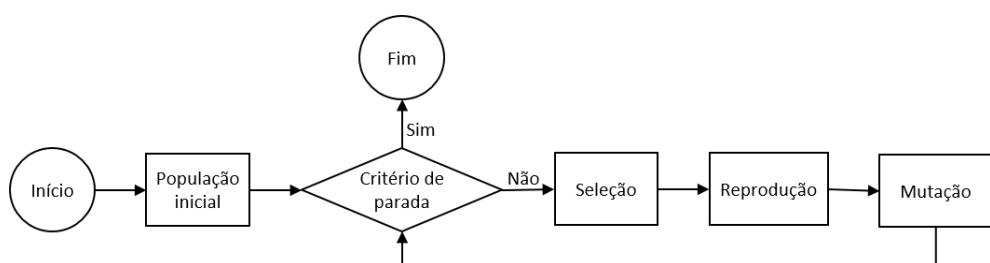
O algoritmo genético inicia com uma população aleatória de indivíduos, representando soluções potenciais. Em seguida, ocorre a etapa de avaliação, na qual cada indivíduo é avaliado em relação ao seu desempenho na função objetivo. Com base nessa avaliação, é atribuída uma medida de aptidão a cada indivíduo, refletindo sua qualidade relativa. Na etapa de seleção, indivíduos mais aptos têm maior probabilidade de serem selecionados para reprodução, com o objetivo de propagar características desejáveis para as próximas gerações.

Na etapa de reprodução do algoritmo genético, indivíduos da população são selecionados com base em sua aptidão para a reprodução. Em seguida, eles são combinados entre si através de operadores genéticos, como cruzamento e recombinação, para gerar uma nova descendência. A nova descendência é influenciada pelas características dos indivíduos pais, mas não necessariamente herda apenas as melhores características da solução anterior.

Além disso, durante a etapa de reprodução, também pode ser aplicada uma mutação aos indivíduos selecionados. A mutação envolve pequenas alterações aleatórias nos seus genes ou variáveis, com o objetivo de explorar novas regiões do espaço de busca.

Para encontrar a solução final, o processo de seleção, reprodução e mutação é repetido por várias gerações ou até que um critério de parada seja atingido. O critério de parada pode ser determinado pela quantidade de gerações, pela convergência da solução, por um limite de tempo, entre outros fatores. A Figura 5 mostra de forma simplificada as etapas do algoritmo genético.

Figura 5 - Fluxograma do algoritmo genético.



2.2 Modelos de aprendizado de máquina

Essa seção descreve alguns algoritmos de regressão do aprendizado de máquina supervisionado.

2.2.1 Regressão Ridge

A regressão *Ridge* é uma técnica de regressão linear, ou seja, ela faz a previsão da saída através da combinação linear das entradas. O algoritmo *Ridge* foi apresentado primeiramente por Hoerl *et al.* (HOERL et al., 1970) com o objetivo de corrigir o algoritmo de regressão tradicional o problema da multicolinearidade. Esse problema acontece quando há uma forte relação linear entre duas ou mais variáveis independentes. Nesse algoritmo, os coeficientes são penalizados em torno do valor zero (e não igual a zero) com o objetivo de selecionar os mais importantes. Dessa forma, nesse algoritmo é adicionado um termo de penalização na função de custo da regressão linear.

Para definir a equação do algoritmo *Ridge* é necessário fazer a definição geral de um algoritmo linear pela equação (6).

$$f(X) = \beta_0 + \sum_{j=1}^p X_j \beta_j \quad (6)$$

onde, p é quantidade de variáveis independentes, β_j são os coeficientes do modelo, β_0 é o termo independente e X_j são as variáveis de entrada. Os β_j 's são desconhecidos, e umas das formas mais populares para estimá-los é usando o método dos mínimos quadrados para minimizar a soma quadrática dos resíduos pela equação (7) (HASTIE et al., 2009).

$$\hat{\beta}^{linear} = argmin_{\beta} \left\{ \sum_{i=1}^N (y_i - f(x_i))^2 = \sum_{i=1}^N \left(y_i - \beta_0 - \sum_{j=1}^p x_{ij} \beta_j \right)^2 \right\} \quad (7)$$

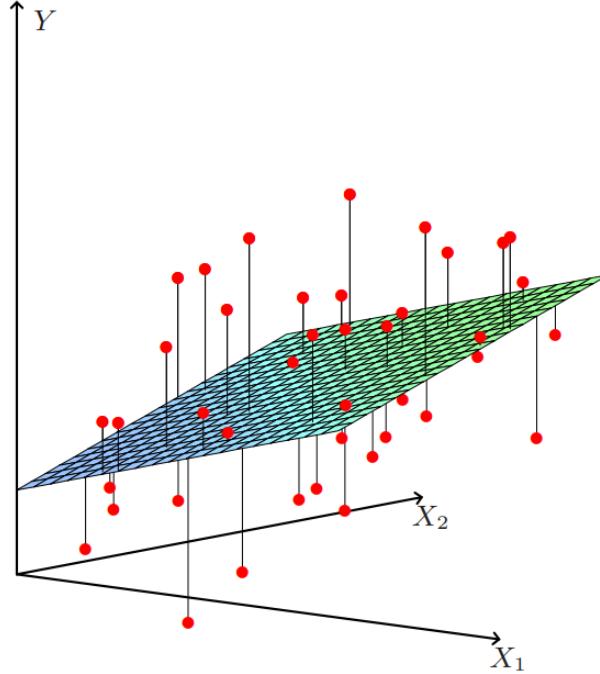
onde y_i é o valor real e $f(x_i)$ é o valor previsto pelo modelo. A Figura 6 mostra um plano ($X \in \mathbb{R}^2$) que minimiza a soma dos resíduos quadrados de Y (valor real).

Ao aplicar o cálculo de minimização de uma função, é possível obter uma única solução no formato matricial, conforme demonstrado na equação (8).

$$\hat{\beta} = (X^T X)^{-1} X^T y \quad (8)$$

onde X são as variáveis de entrada, X^T é a transposta de X e y é o rótulo desejado.

Figura 6 - Ajuste dos mínimos quadrados com $X \in \mathbb{R}^2$.



Fonte: (HASTIE et al., 2009).

Dessa forma, o valor previsto pelo modelo pode ser escrito como na equação (9). Onde X é formado x_0, x_1, \dots, x_p , com $x_0 = 1$ e $\hat{\beta}$ é composto por $\beta_0, \beta_1, \dots, \beta_p$.

$$\hat{y} = X\hat{\beta} = X(X^T X)^{-1} X^T y \quad (9)$$

Conforme mencionado anteriormente, na regressão *Ridge* é adicionado um termo de penalização $\lambda \geq 0$, o que modifica a equação (7) para a equação (10).

$$\hat{\beta}^{ridge} = \operatorname{argmin}_{\beta} \left\{ \sum_{i=1}^N \left(y_i - \beta_0 - \sum_{j=1}^p x_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^p (\beta_j)^2 \right\} \quad (10)$$

A solução para minimização de $\hat{\beta}^{ridge}$ é expressa pela equação (11), enquanto a equação (12) é utilizada para calcular o valor previsto pelo modelo.

$$\hat{\beta}^{ridge} = (X^T X + \lambda I)^{-1} X^T y \quad (11)$$

$$\hat{y} = X\hat{\beta} = X(X^T X + \lambda I)^{-1} X^T y. \quad (12)$$

Pode-se observar que houve o acréscimo do termo λI , que causa uma perturbação na diagonal principal de $X^T X$ devido à penalização quadrática (VASCONCELOS, 2017). Como consequência, nenhuma variável é excluída, já que o método diminui a magnitude dos coeficientes, mas sem defini-los iguais a zero. Dessa forma, mesmo que a variável não contribua significativamente para a previsão, ela ainda estará presente nos coeficientes com um valor muito baixo, o que permite sua remoção posterior sem afetar o desempenho do algoritmo.

2.2.2 Regressão Lasso

A regressão *Lasso*, assim como a regressão *Ridge*, também é um método com penalização nos coeficientes. Por esse motivo, esses métodos são conhecidos como *shrinkage*. No entanto, diferentemente do *Ridge*, o termo de penalização do *Lasso* não está ao quadrado, como mostra a equação (13). Em vez disso, a penalidade do *Ridge* ($\lambda \sum_{j=1}^p (\beta_j)^2$) é substituída pela penalidade do *Lasso* ($\sum_{j=1}^p |\beta_j|$) (HASTIE et al., 2009).

$$\hat{\beta}^{lasso} = \operatorname{argmin}_{\beta} \left\{ \sum_{i=1}^N \left(y_i - \beta_0 - \sum_{j=1}^p x_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^p |\beta_j| \right\} \quad (13)$$

A solução do problema de otimização *Lasso* é mais complexa que a regressão *Ridge*, pois esse tipo de penalização torna a solução para y_i não linear. Dessa forma, não há uma expressão fechada como na equação (11). Para encontrar uma solução é necessário usar técnicas de programação quadrática (VASCONCELOS, 2017).

Diferentemente da regressão *Ridge*, a regressão *Lasso* permite que alguns coeficientes sejam iguais a zero. Com isso, é feito um processo de seleção de variáveis que permite diminuir a dimensão do problema e melhorar a interpretabilidade do modelo. Porém, ao contrário da regressão *Ridge*, a regressão *Lasso* tende a ignorar o problema de multicolinearidade das variáveis (VASCONCELOS, 2017).

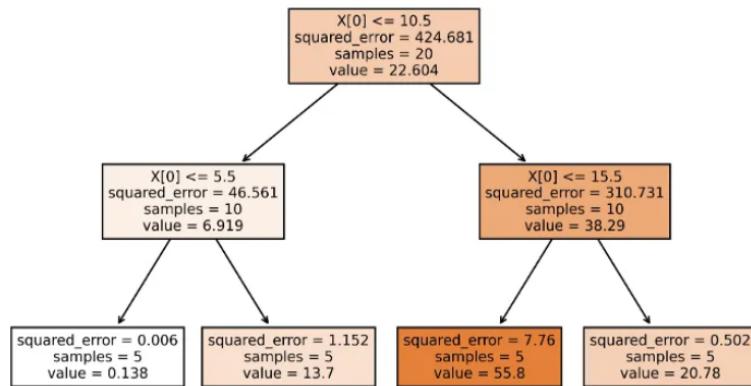
2.2.3 Regressão Random Forest

Para entender o funcionamento do método *Random Forest* (RF), é necessário definir primeiramente a Árvore de Decisão (*Decision Tree* - DT). A DT pode ser usada em

algoritmos de classificação e de regressão. Neste trabalho, a descrição será limitada à regressão.

A DT é um algoritmo de aprendizado de máquina não linear, representado por uma árvore. A essência da DT é dividir a amostra de treinamento em subespaços, de forma que o erro quadrático seja mínimo. Dessa maneira, o algoritmo escolhe o melhor atributo e a regra para divisão do subconjunto. Esse processo continua até que a condição de parada seja atendida. A Figura 7 mostra um exemplo numérico de uma árvore de decisão, no qual o nó raiz é o ponto de partida da divisão e os últimos nós (folhas) são os resultados (saída) do algoritmo. Pode-se observar que, neste exemplo, o erro quadrático está diminuindo em cada nível da árvore. O algoritmo de DT é de fácil interpretação, ajuda na identificação dos atributos mais importantes na divisão e é mais indicado quando as relações entre as variáveis são complexas.

Figura 7 - Regressão na árvore de decisão.



Fonte: (SANTOS, 2022).

Segundo Hastie *et al.* (HASTIE et al., 2009), a equação (14) da regressão em árvore é similar à regressão linear.

$$f(X) = \sum_{m=1}^M c_m I(x \in \mathbb{R}_m) \quad (14)$$

onde, R_1, \dots, R_m são partições do espaço. A $I(x \in \mathbb{R}_m)$ é uma função de indicação que assume o valor um quando $X \in \mathbb{R}_m$ e zero caso contrário. Já c_m é apenas a média de y_i na região R_m , ou seja, $\hat{c}_m = \text{avg}(y_i | x_i \in \mathbb{R}_m)$.

Para construir o algoritmo de regressão em árvore, é necessário minimizar a soma do quadrado dos erros da previsão de cada subconjunto. Porém, encontrar a melhor divisão binária em termos dos mínimos quadrados é inviável computacionalmente. Dessa forma, será usado um algoritmo guloso e os dados serão divididos por uma variável j , um ponto de divisão s e um par de semiplanos representados pela equação (15) (HASTIE et

al., 2009).

$$R_1(j, s) = \{X | X_j \leq s\} \quad \text{e} \quad R_2(j, s) = \{X | X_j > s\} \quad (15)$$

Em seguida, são buscados as variáveis j e s que resolvem a equação (16). Para qualquer escolha j e s , a minimização interna é resolvida por $\hat{c}_1 = \text{ave}(y_i | x_i \in R_1)$ e $\hat{c}_2 = \text{avg}(y_i | x_i \in R_2)$.

$$\min_{j,s} \left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right] \quad (16)$$

Após encontrar a melhor divisão, os dados são particionados em duas regiões e o processo de divisão é repetido. Isso pode ser visto no exemplo da Figura 7, na qual o nó raiz foi se dividindo até alcançar as folhas. Porém, a equação (16) não tem um critério de parada bem definido. Por isso, é necessário usar abordagens que limitam a profundidade máxima da árvore. Caso contrário, o modelo sofrerá um *overfitting*, o que indica que o modelo se adaptou demais aos dados de treinamento, resultando em um desempenho insatisfatório na previsão de novos dados e não alcançando a generalização desejada.

Um outro problema encontrado na DT é que há uma sensibilidade a mudanças na amostra de treinamento, e isso pode prejudicar o desempenho do modelo. Dessa forma, existem alguns métodos que melhoraram o desempenho preditivo, como o *Random Forest* (VASCONCELOS, 2017).

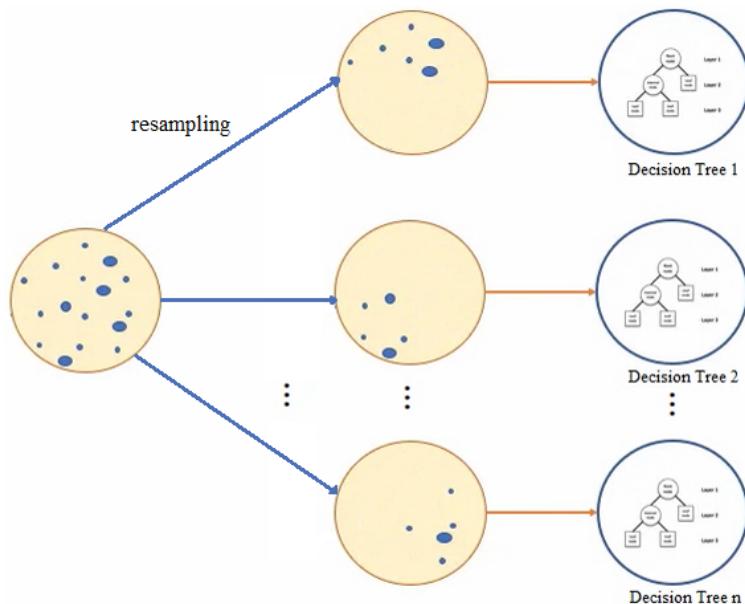
O *Random Forest* é um algoritmo de *bagging*, ou seja, consiste em um grande número de DT individuais que fazem uma amostragem aleatória do conjunto de dados de treino com reposição (*bootstrap*), o que resulta em árvores diferentes. Cada árvore individual faz uma previsão, e o resultado final do modelo será a média de cada uma das árvores. A Figura 8 mostra de forma simplificada a estrutura do algoritmo *bagging*.

O algoritmo de DT no momento de dividir um nó, considera todas as variáveis possíveis e escolhe aquela com menor erro residual. Já o algoritmo *Random Forest* escolhe apenas um subconjunto aleatório das entradas. Isso faz com que as árvores tenham maior variação, resultando em uma menor correlação entre elas e melhorando o desempenho do modelo.

2.2.4 Regressão Light Gradient Boosting Machine

O modelo *Light Gradient Boosting Machine* (LightGBM) é um modelo de aprendizado de máquina que usa a estratégia *boosting* com uso da DT. Nessa estratégia, é criado

Figura 8 - Esquema simplificado para algoritmo *bagging* usado no *Random Forest*.



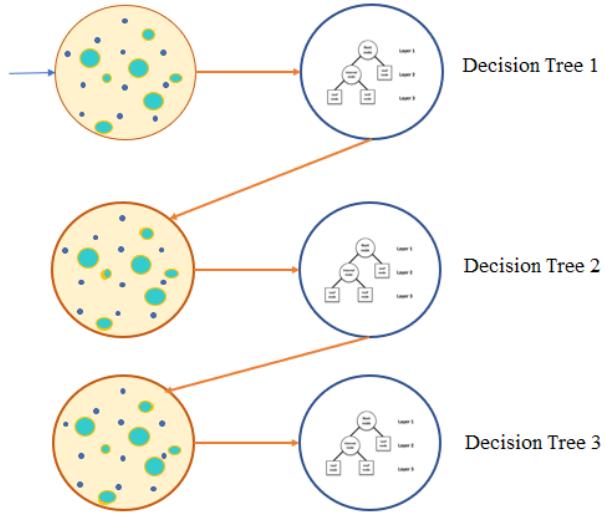
Fonte: (KHILLAR, 2019) (Adaptado).

um novo modelo de DT em cada iteração. Cada iteração consiste em combinar vários modelos de Árvore de Decisão sequencialmente, em que a entrada de cada modelo é ajustada com base nos pesos atualizados pelos modelos anteriores. O *boosting* atribui um peso igual para cada amostra de dados para o primeiro modelo (*baseline*), já os demais pesos são ajustados conforme o erro. Dessa forma, cada nova árvore é treinada para corrigir os erros da árvore anterior. Para o cálculo do erro, uma parte do conjunto de treinamento que não é vista pelo modelo é utilizada, e esse conjunto é chamado de conjunto de validação. O modelo final é uma média ponderada dos modelos predecessores. A Figura 9 mostra de forma simplificada a estrutura do método *boosting*.

As características citadas são semelhantes ao modelo Gradient Boosting Decision Tree (GBDT). Porém, o LightGBM é uma forma otimizada. Segundo Ke *et al.* (KE et al., 2017), o LightGBM acelera o processo de treinamento do GBDT em até 20 vezes. Para esse ganho de desempenho, foram implementadas duas técnicas: Gradient-based One-Side Sampling (GOSS) e Exclusive Feature Bundling (EFB).

O GOSS exclui uma proporção significativa de instâncias de dados (linhas) com pequenos gradientes, e deixa apenas o restante para estimar o ganho de informação. As instâncias de dados com gradientes maiores desempenham um papel mais importante no cálculo do ganho de informação. Com isso, o GOSS pode obter uma estimativa bastante precisa do ganho de informação com um tamanho de dados muito menor. O EFB agrupa as *features* (colunas) que são mutuamente exclusivas, ou seja, que não assumem valores diferentes de zero simultaneamente, a fim de reduzir o número de *features*. Para encontrar

Figura 9 - Esquema simplificado para algoritmo *boosting* usado no LightGBM.



Fonte: (KHILLAR, 2019) (Adaptado).

o agrupamento ideal, é utilizado um algoritmo guloso. (KE et al., 2017).

Portanto, o LightGBM é adequado para conjunto de dados grandes e com muitas variáveis, já que possui técnicas de otimização como GOSS e EFB.

2.3 Métricas de avaliação

Nas seções anteriores, foi citada a melhoria no desempenho do modelo. Essa melhoria pode estar relacionada ao tempo de execução do algoritmo ou à capacidade do modelo de fazer previsões corretas. Nesse sentido, são usadas métricas de avaliação para encontrar o modelo que possui o menor erro na previsão e qual se ajusta melhor aos dados. Existem diversas métricas, mas neste trabalho serão descritas três métricas de avaliação em modelos de regressão, sendo elas:

- Coeficiente de determinação (R^2):** É interpretado como a proporção da variável dependente que é previsível a partir das variáveis independentes (CHICCO et al., 2021). É calculado pela equação (17). Quanto mais próximo de 1 melhor é o modelo.

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (17)$$

Onde n é quantidade de dados, y_i é o valor real, \hat{y}_i é o valor previsto e \bar{y}_i é a média dos dados.

2. **Erro Absoluto Médio (Mean Absolute Error - MAE)**: É a média das distâncias entre os valores previstos e os reais descrita pela equação (18). Ou seja, informa qual o erro médio do modelo na unidade dos dados previstos. Quanto mais próximo de zero for essa métrica, melhor será o modelo.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |\hat{y}_i - y_i| \quad (18)$$

3. **Raiz do Erro Quadrático Médio (Root Mean Squared Error - RMSE)**: É a raiz quadrada da média dos quadrados dos erros da previsão, conforme a equação (19). Assim como o MAE, essa métrica também informa o erro do modelo na unidade dos dados. A diferença é que o RMSE amplifica os erros, ou seja, os erros que são grandes tornam-se maiores. Quanto mais próximo de zero for essa métrica, melhor será o modelo.

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2} \quad (19)$$

2.4 Busca de hiperparâmetros

Os hiperparâmetros são parâmetros ajustáveis antes do treinamento do modelo de aprendizado de máquina. A DT possui alguns hiperparâmetros, como a profundidade máxima da árvore, o critério para medir a qualidade de uma divisão (que pode ser o erro quadrático ou erro absoluto) e o número mínimo de amostras para dividir um nó. Todos esses hiperparâmetros influenciam no desempenho do modelo, por isso surge a dúvida: qual o valor usar? Não existe um valor fixo/ideal para os hiperparâmetros, já que os dados mudam de um problema para o outro. Dessa forma, o desempenho do modelo irá mudar para diferentes valores de hiperparâmetros. Por isso, existem algoritmos que buscam quais são os valores ideais dos hiperparâmetros para o modelo ter o melhor resultado. Como, por exemplo, *Grid Search*, *Random Search* e a busca Bayesiana.

1. ***Grid Search***: Esse método testa todas as combinações possíveis de valores, dentro de um conjunto, até encontrar os melhores. Ou seja, é um algoritmo de força bruta que testa todas as possibilidades até encontrar a melhor. Essa busca pode levar muito tempo caso a quantidade de hiperparâmetros seja grande.
2. ***Random Search***: Esse método é semelhante ao *Grid Search*, porém, a busca dos hiperparâmetro dentro do conjunto de valores é feita aleatoriamente. Dessa forma,

não é necessário testar todas as combinações possíveis, o que torna esse método mais eficiente.

3. **Busca Bayesiana:** Esse algoritmo de busca usa um modelo probabilístico para estimar a função objetivo. Essa função é responsável por avaliar o desempenho do modelo probabilístico em relação aos hiperparâmetros selecionados. A busca Bayesiana inicia com uma distribuição de probabilidade a priori que é atualizada a cada iteração (usando Teorema de Bayes) a fim de avaliar os valores de hiperparâmetros escolhidos e também escolher a próxima combinação deles (JÚNIOR, 2018). Esse método é mais eficiente do que os dois métodos anteriores quando se lida com um grande número de hiperparâmetros.

2.5 Aprendizado de máquina automatizado

Aprendizado de Máquina Automatizado (*Automated Machine Learning - AutoML*) é o processo de automatizar as tarefas no desenvolvimento de um modelo de aprendizado de máquina (GOSWAMI, 2023). Geralmente essas tarefas são realizadas por *frameworks*.

Todas as etapas no desenvolvimento de um modelo, como a transformação dos dados, a divisão em treino e teste, testar hiperparâmetros, executar o modelo e avaliar as métricas ficam sob responsabilidade do *framework*. Dessa forma, é necessário que o usuário defina apenas se o modelo é de regressão ou classificação, as entradas e saídas, bem como a métrica principal de avaliação do modelo. Com esses passos, o *framework* testa diversos modelos de aprendizado de máquina e informa uma lista com os modelos testados, e ainda qual é o melhor com base na métrica escolhida anteriormente.

Uma das vantagens do uso do AutoML é a possibilidade de criar rapidamente um modelo *baseline* capaz de resolver o problema em questão. Entretanto, uma das desvantagens é que nem sempre os resultados são satisfatórios e é difícil identificar se ocorreu *overfitting*, uma vez que a interpretação do modelo pode ser comprometida.

3 TRABALHOS RELACIONADOS

A previsão da RUL do motor Turbofan tem sido amplamente estudada. No artigo (HOMEM et al., 2020), foram utilizadas técnicas de pré-processamento, como a normalização padrão dos dados, retirada de dados constantes dos sensores (*flat*) e a junção dos conjuntos de treino e teste, seguida de uma separação aleatória de 30% dos dados para fins de teste. Além disso, os autores empregaram uma técnica proposta por Heimes (HEIMES, 2008), que assume a RUL como constante durante os ciclos iniciais. Esses ciclos iniciais foram definidos pelos autores como aqueles em que a RUL é superior a 130. O modelo proposto no artigo foi de uma rede neural com filtros convolucionais e camadas recorrentes do tipo *Long Short Term Memory* (LSTM). O modelo teve um bom desempenho apresentando RMSE em torno de 19.

Três outros estudos seguiram um processo semelhante ao artigo anterior, mas com conjuntos de dados de treinamento e teste separados. No artigo (ELLEFSEN et al., 2019), os autores desenvolveram um modelo de aprendizado profundo semi-supervisionado com base em máquinas de Boltzmann restritas e LSTM, que teve uma média de RMSE em torno de 17.

Em contrapartida, o segundo trabalho (BABU et al., 2016) adotou uma abordagem de regressão baseada em redes neurais convolucionais profundas para estimar a RUL, obtendo um RMSE de 18,75.

No artigo (PENG et al., 2021), os autores realizaram um pré-processamento semelhante aos dois trabalhos anteriores, no entanto, eles consideraram os ciclos iniciais para RUL maior que 125 e selecionaram variáveis. O modelo proposto pelos autores é uma combinação de redes neurais convolucionais unidimensionais com camada convolucional completa (1-FLCCNN) e LSTM, que alcançou um RMSE igual a 11,07.

Além dos trabalhos com modelos em redes neurais, a literatura apresenta artigos em que os autores comparam modelos como árvores de decisão, modelos lineares e algoritmos de boosting. No artigo (YUREK; BIRANT, 2019), são comparados diversos modelos como: *Linear Regression*, *Bayesian Linear Regression*, *Poisson Regression*, *Bosted Decision Tree Regression* e *Decision Forest Regression*. Os autores usaram dois tipos de pré-processamento. O primeiro utiliza a média, desvio padrão e outros métodos estatísticos. Já o segundo, foi calculado uma *feature* categórica para classificar o risco como baixa, média e alta. Essa classificação foi feita com base no tempo de operação do motor e no desvio padrão das *features*. O melhor modelo desenvolvido pelos autores foi *Decision Forest Regression* com MAE igual a 12,30.

No artigo (MATHEW et al., 2017), vários modelos também foram testados, porém, não foram fornecidos detalhes sobre o pré-processamento empregado. Entre os modelos avaliados, o *Random Forest* apresentou o melhor desempenho com RMSE média igual a

29,73.

4 PREPARAÇÃO DOS DADOS

A preparação dos dados consiste no processo de transformação e limpeza dos dados brutos, a fim de prepará-los em um formato adequado para o treinamento do modelo de aprendizado de máquina. Essa etapa abrange diversas atividades, como extração dos dados, análise exploratória e pré-processamento. É considerada uma fase fundamental, uma vez que a qualidade dos dados influencia diretamente na qualidade do modelo resultante.

Nas seções a seguir, serão detalhadas todas as etapas de preparação dos dados realizadas na base de dados utilizada neste trabalho. Caso deseje obter informações detalhadas de como foram desenvolvidas todas as etapas do trabalho, é possível acessar o código fonte do projeto no GitHub¹.

4.1 Extração dos dados brutos

A extração dos dados brutos consiste na coleta diretamente da fonte sem qualquer processamento ou transformação nos dados. Como foi citado na Seção 1.3, a base de dados utilizada no projeto foi do motor Turbofan. Essa base de dados é composta por quatro *datasets* (FD001, FD002, FD003 e FD004) correspondentes a cada motor diferente. Os dois primeiros possuem os dados do modo de falha do HPC, e os dois últimos possuem os dados do HPC e da *fan*. O modo de falha é a forma pela qual o motor pode falhar em desempenhar sua função, podendo ser ocasionado por diversos fatores. Esses quatro motores apresentam diferentes níveis de desgaste inicial, porém todos começam operando em condição normal, sem indicação de falha. Os *datasets* ímpares (FD001 e FD003) compartilham o mesmo nível de desgaste inicial, e o mesmo acontece com os conjuntos de dados pares (FD002 e FD004).

Cada um dos *datasets* é dividido em subconjuntos de treino e teste, os quais consistem em diversas séries temporais medidas em ciclos. O conjunto de treino é utilizado para treinar um modelo de aprendizado de máquina, e é composto por informações que permitem ao modelo aprender a relação entre as entradas e as saídas esperadas. Já o conjunto de teste é usado para avaliar o desempenho do modelo após o treinamento. Esse conjunto é composto por dados que o modelo nunca viu antes e que compartilham as mesmas características do conjunto de treino. Essa divisão é fundamental porque um modelo pode ser capaz de se ajustar muito bem aos dados de treino, mas pode não ser capaz de prever com um bom desempenho dados que nunca foram vistos antes.

¹ Código fonte do trabalho: https://github.com/marcelov-aguiar/pdm_turbofan

Durante os ciclos, o motor opera inicialmente em condição normal e eventualmente desenvolve uma falha em algum ponto. No conjunto de treinamento, a degradação evolui gradualmente até a falha do motor. Já no conjunto de teste, o ciclo é interrompido em um determinado momento antes da falha do motor. Na seção 4.2 será explicado como identificar a falha.

O conjunto de dados utilizado neste trabalho é de acesso público e foi disponibilizado pela NASA em um arquivo de texto contendo 26 colunas separadas por espaços. Os arquivos de treinamento e teste contêm aproximadamente 34 mil linhas cada (TEUBERT, 2007). É importante salientar que os dados já estão pré-separados em conjuntos distintos de treinamento e teste. A Tabela 2 apresenta informações sobre as *features* dos *datasets*, incluindo nome, descrição e unidade de medida.

4.2 Análise exploratória

A análise exploratória é uma abordagem fundamental na análise de dados, que envolve o uso de técnicas estatísticas e visualização de dados para identificar padrões, tendências, correlações e anomalias. Seu principal objetivo é descobrir *insights* e revelar informações ocultas nos dados, fornecendo subsídios para a tomada de decisões na etapa de pré-processamento. Ao aplicar técnicas estatísticas e visualizações, é possível obter uma compreensão mais profunda dos dados e extrair conhecimentos relevantes para direcionar as próximas etapas da análise. (FACELI et al., 2011).

A análise dos *datasets* foi realizada com a linguagem de programação Python em conjunto com as bibliotecas de análise de dados (Numpy e Pandas) e visualização (Matplotlib e Seaborn). A análise foi conduzida em duas etapas distintas. A primeira etapa foi dedicada à limpeza do conjunto de dados, enquanto a segunda foi responsável por identificar *insights* e transformações nos dados que poderiam aprimorar as métricas do modelo.

1. Primeira etapa

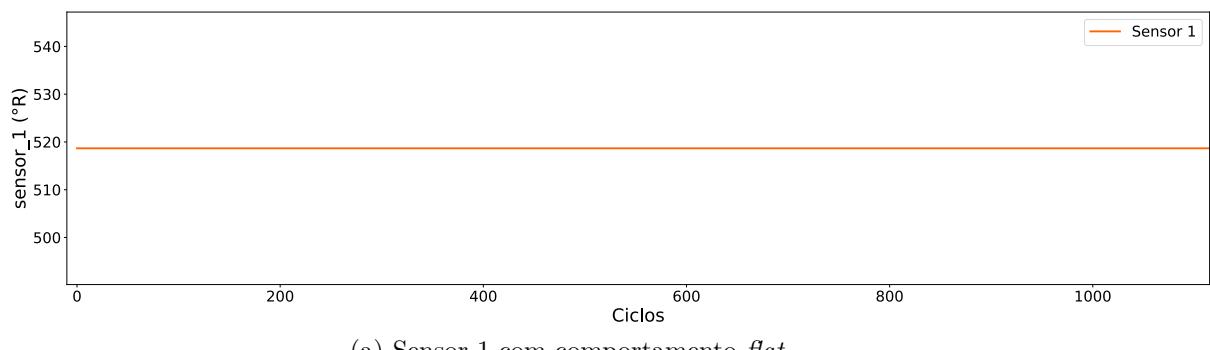
Após realizar a primeira etapa da análise, concluiu-se que a limpeza dos dados não era necessária, uma vez que não foram identificados dados nulos, *outliers* ou dados duplicados. Entretanto, foi necessário excluir sensores com valores *flat*, ou seja, com variância próxima de zero. O Gráfico 1 mostra um sensor ruim (*flat*) e o outro bom.

O Gráfico 1b ainda apresenta o conceito de campanha. Observa-se que o valor do sensor 2 apresenta um aumento gradual até um determinado momento, seguido de uma queda brusca, que indica a ocorrência de uma falha no motor. Em seguida, o motor é submetido à manutenção e, após seu reparo, volta a operar normalmente, como evidenciado pela retomada do crescimento da curva no gráfico. Nesse con-

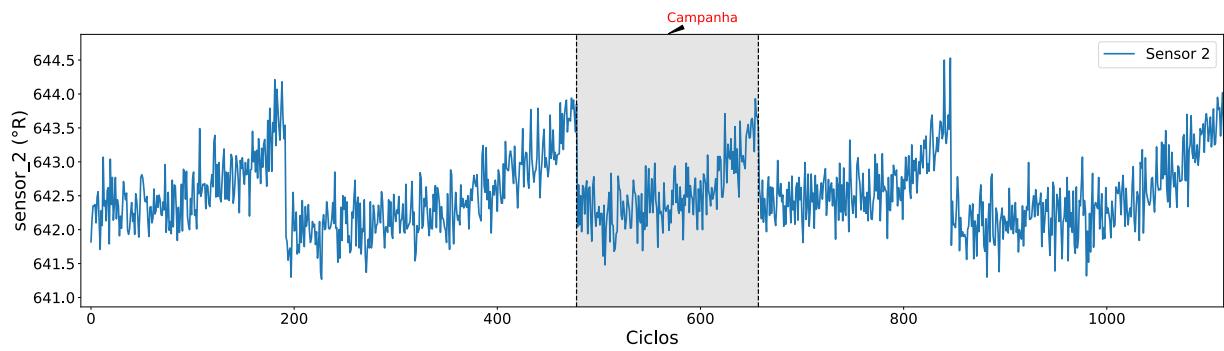
Tabela 2 - Descrição dos dados dos sensores do Turbofan.

Nome da <i>feature</i>	Descrição	Unidade
Unit number	É o número da unidade do motor. Assume valores de 1 a 100	-
Time	É o tempo que o motor está em operação	ciclos
Operational setting 1	Tipo de configuração do motor	-
Operational setting 2	Tipo de configuração do motor	-
Operational setting 3	Tipo de configuração do motor	-
Sensor 1	Temperatura de entrada da <i>fan</i>	°R
Sensor 2	Temperatura de saída do LPC	°R
Sensor 3	Temperatura de saída do HPC	°R
Sensor 4	Temperatura de saída da LPT	°R
Sensor 5	Pressão de entrada da <i>fan</i>	psia
Sensor 6	Pressão do ducto de desvio	psia
Sensor 7	Pressão de saída do HPC	psia
Sensor 8	Velocidade física da <i>fan</i>	rpm
Sensor 9	Velocidade física do núcleo	rpm
Sensor 10	Razão de pressão (P50/P2)	-
Sensor 11	Pressão estática de saída do HPC	psia
Sensor 12	Razão da vazão do combustível para Ps30	pps/psia
Sensor 13	Velocidade corrigida da <i>fan</i>	rpm
Sensor 14	Velocidade corrigida do núcleo	rpm
Sensor 15	Razão de desvio	-
Sensor 16	Relação ar-combustível do combustor	-
Sensor 17	Drenagem de entalpia	-
Sensor 18	Velocidade necessária da <i>fan</i>	rpm
Sensor 19	Velocidade de conversão necessária da <i>fan</i>	rpm
Sensor 20	Fluxo de ar frio da HPT	lb/s
Sensor 21	Fluxo de ar frito da LPT	lb/s

Gráfico 1 - Comparação dos sensores 1 e 2 em relação aos ciclos.



(a) Sensor 1 com comportamento *flat*.



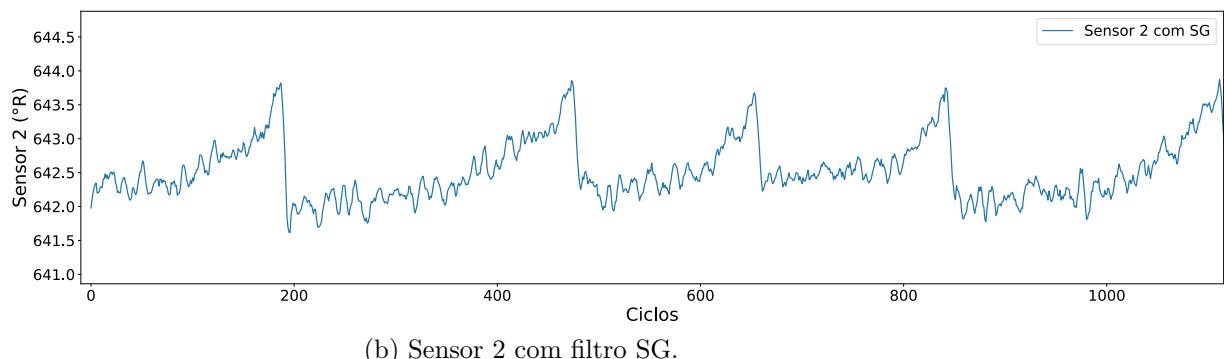
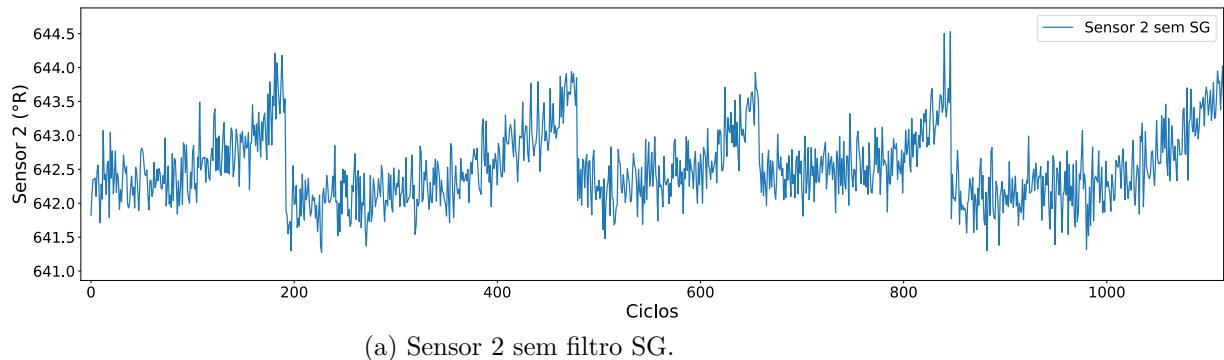
(b) Sensor 2 com comportamento normal.

texto, a campanha pode ser definida como o período em que o motor permanece em operação entre intervalos de manutenção.

2. Segunda etapa

Na segunda etapa, verificou-se necessidade de calcular a RUL, a partir da *feature time*, e realizar a normalização padrão nos dados. Também foi levantada a hipótese de aplicar uma suavização nos dados dos sensores, já que eles apresentam elevado nível de ruído. O Gráfico 2 ilustra a diferença entre o sensor 2 sem e com a aplicação do filtro SG, com uma janela de tamanho 2 e um polinômio de segundo grau. Já o Gráfico 2b evidencia a redução do ruído nos dados após a suavização.

Gráfico 2 - Comparaçāo dos dados do sensor 2 antes e apōs a aplicāo do filtro SG.



Uma outra hipótese a ser considerada é que os dados e a degradação do motor Turbofan possuem características não lineares. Por isso, é provável que modelos não lineares apresentem melhor desempenho em comparação com modelos lineares.

4.3 Pré-processamento

O pré-processamento de dados prepara os dados brutos para utilização nos modelos de aprendizado de máquina. Nessa etapa, são aplicadas as técnicas de transformação de dados mencionadas na Seção 2.1, a fim de melhorar a qualidade e relevância dos dados. É no pré-processamento que os *insights* e hipóteses levantados durante a análise exploratória são implementados e testados, com o objetivo de obter melhores resultados na modelagem. Inicialmente foram feitos os seguintes passos nos *datasets* do Turbofan:

- 1. Remoção das *features* com variância próxima a zero:** Algumas *features* com baixa variância foram removidas, como o *sensor_1*.
- 2. Padronização da estrutura de dados:** Nessa etapa, foi realizada a nomeação dos sensores e a generalização do código para permitir a execução em todos os quatro *datasets* disponíveis. Para garantir a padronização da estrutura dos dados, foi necessário convertê-los em um *DataFrame* da biblioteca Pandas. Ela fornece estruturas de dados de alto desempenho e fáceis de usar, como o *DataFrame*, que permite a manipulação e análise de dados tabulares. Com essa biblioteca, é possível carregar, manipular, transformar e combinar dados de diferentes fontes, além de realizar operações estatísticas e de modelagem de dados.

A Figura 10 ilustra a estrutura de dados do *DataFrame* utilizado para armazenar os dados do conjunto de treinamento. É possível notar que as *features* são representadas em colunas no *DataFrame*, e essas colunas são as entradas para o modelo de aprendizado de máquina (exceto pela *feature unit_number*). Cada linha corresponde a um ciclo, que é registrado pela *feature time*, sendo que cada ciclo corresponde a um voo do avião. Além disso, o *DataFrame* atribui automaticamente um índice a cada linha, mas essa informação não é usada pelo modelo.

Figura 10 - Estrutura de dados *DataFrame* para os dados de treinamento do *dataset* FD001.

unit_number	time	setting_1	setting_2	sensor_2	sensor_3	sensor_4	sensor_6	sensor_7	sensor_8	sensor_9	
0	1.0	1.0	-0.0007	-0.0004	641.82	1589.70	1400.60	21.61	554.36	2388.06	9046.19
1	1.0	2.0	0.0019	-0.0003	642.15	1591.82	1403.14	21.61	553.75	2388.04	9044.07
2	1.0	3.0	-0.0043	0.0003	642.35	1587.99	1404.20	21.61	554.26	2388.08	9052.94
3	1.0	4.0	0.0007	0.0000	642.35	1582.79	1401.87	21.61	554.45	2388.11	9049.48
4	1.0	5.0	-0.0019	-0.0002	642.37	1582.85	1406.22	21.61	554.00	2388.06	9055.15
...	
20626	100.0	196.0	-0.0004	-0.0003	643.49	1597.98	1428.63	21.61	551.43	2388.19	9065.52
20627	100.0	197.0	-0.0016	-0.0005	643.54	1604.50	1433.58	21.61	550.86	2388.23	9065.11
20628	100.0	198.0	0.0004	0.0000	643.42	1602.46	1428.18	21.61	550.94	2388.24	9065.90
20629	100.0	199.0	-0.0011	0.0003	643.23	1605.26	1426.53	21.61	550.68	2388.25	9073.72
20630	100.0	200.0	-0.0032	-0.0005	643.85	1600.38	1432.14	21.61	550.79	2388.26	9061.48

20631 rows × 19 columns

3. Normalização padrão dos dados: Para garantir que as diferentes magnitudes das *features* não afetem o modelo de aprendizado de máquina, foi realizada a normalização padrão dos dados usando o método *StandardScaler* da biblioteca *scikit-learn* (PEDREGOSA et al., 2011). O *scikit-learn* é uma biblioteca amplamente utilizada para aprendizado de máquina em Python, que oferece diversas funcionalidades para modelagem de dados, pré-processamento e avaliação de modelos, além de ser conhecida por sua facilidade de uso e eficiência computacional. Os modelos *Lasso*, *Ridge* e *Random Forest* utilizados neste trabalho foram implementados a partir dessa biblioteca.

O método *StandardScaler* aplica a equação mencionada na Subseção 2.1.1 aos dados, transformando-os de modo que a média seja zero e o desvio padrão seja unitário. A Figura 11 mostra o *DataFrame* após a normalização, onde é possível notar que as magnitudes das *features* foram normalizadas, tornando-as comparáveis entre si.

Figura 11 - Dados normalizados para o conjunto de treinamento do *dataset* FD001.

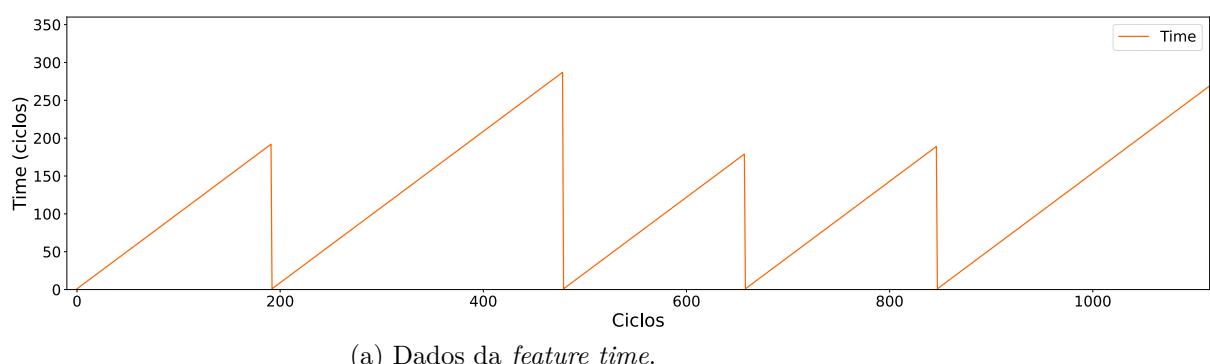
	unit_number	time	setting_1	setting_2	sensor_2	sensor_3	sensor_4	sensor_6	sensor_7	sensor_8	sensor_9
0	-1.728084	-1.565170	-0.315980	-1.372953	-1.721725	-0.134255	-0.925936	0.141683	1.121141	-0.516338	-0.862813
1	-1.728084	-1.550652	0.872722	-1.031720	-1.061780	0.211528	-0.643726	0.141683	0.431930	-0.798093	-0.958818
2	-1.728084	-1.536134	-1.961874	1.015677	-0.661813	-0.413166	-0.525953	0.141683	1.008155	-0.234584	-0.557139
3	-1.728084	-1.521616	0.324090	-0.008022	-0.661813	-1.261314	-0.784831	0.141683	1.222827	0.188048	-0.713826
4	-1.728084	-1.507098	-0.864611	-0.690488	-0.621816	-1.251528	-0.301518	0.141683	0.714393	-0.516338	-0.457059
...
20626	1.659204	1.265868	-0.178822	-1.031720	1.618000	1.216258	2.188375	0.141683	-2.189329	1.315066	0.012547
20627	1.659204	1.280386	-0.727453	-1.714186	1.717992	2.279706	2.738351	0.141683	-2.833345	1.878576	-0.006020
20628	1.659204	1.294904	0.186933	-0.008022	1.478011	1.946971	2.138377	0.141683	-2.742957	2.019453	0.029755
20629	1.659204	1.309423	-0.498857	1.015677	1.098043	2.403666	1.955051	0.141683	-3.036719	2.160330	0.383884
20630	1.659204	1.323941	-1.458962	-1.714186	2.337940	1.607712	2.578358	0.141683	-2.912435	2.301208	-0.170405

20631 rows × 19 columns

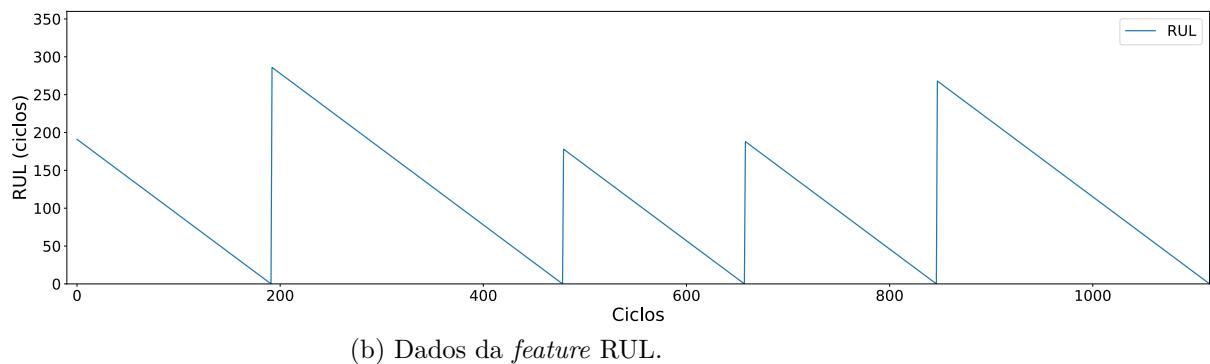
4. Cálculo da RUL: Como mencionado anteriormente, a RUL foi calculada a partir da *feature time*, uma vez que essa variável não estava presente nos dados brutos. A RUL é obtida pela inversão da *feature time* em cada campanha. O Gráfico 3a mostra a *feature time* aumentando com o passar dos ciclos, ou seja, indicando quantos ciclos se passaram desde o início da campanha. Já o Gráfico 3b exibe a curva resultante da RUL, que decresce até atingir zero, momento em que ocorre a falha do motor. Portanto, a RUL indica quantos ciclos restam para a falha acontecer. Assim, o objetivo do modelo é prever a curva da RUL por meio dos dados históricos das *features* de entrada.

Após calcular a RUL, foi possível gerar um gráfico da matriz de correlação de Pearson. Essa matriz é uma medida estatística que avalia a relação linear entre duas variáveis contínuas. Na matriz de correlação de Pearson cada célula representa o coeficiente de correlação entre duas variáveis (LIDIANE et al., 2018). Esse coeficiente varia entre -1 e 1, indicando a intensidade e a direção da relação linear entre

Gráfico 3 - Comparação entre as curvas *time* e RUL para o *dataset* FD001.



(a) Dados da *feature time*.



(b) Dados da *feature RUL*.

as variáveis, sendo que:

- 1 indica uma correlação perfeitamente positiva
- 0 indica nenhuma correlação
- -1 indica uma correlação perfeitamente negativa

O objetivo da análise da correlação de Pearson é identificar as variáveis que podem ter um impacto maior nos coeficientes dos modelos lineares. Embora uma alternativa seja o coeficiente de Spearman, que permite uma correlação não linear, neste trabalho foi realizado apenas a correlação de Pearson para avaliar os coeficientes dos modelos lineares.

Os coeficientes da correlação de Pearson são obtidos por meio da equação (20), em que x e y representam duas variáveis quaisquer, e \bar{x} e \bar{y} são as médias amostrais dos conjuntos de dados correspondentes.

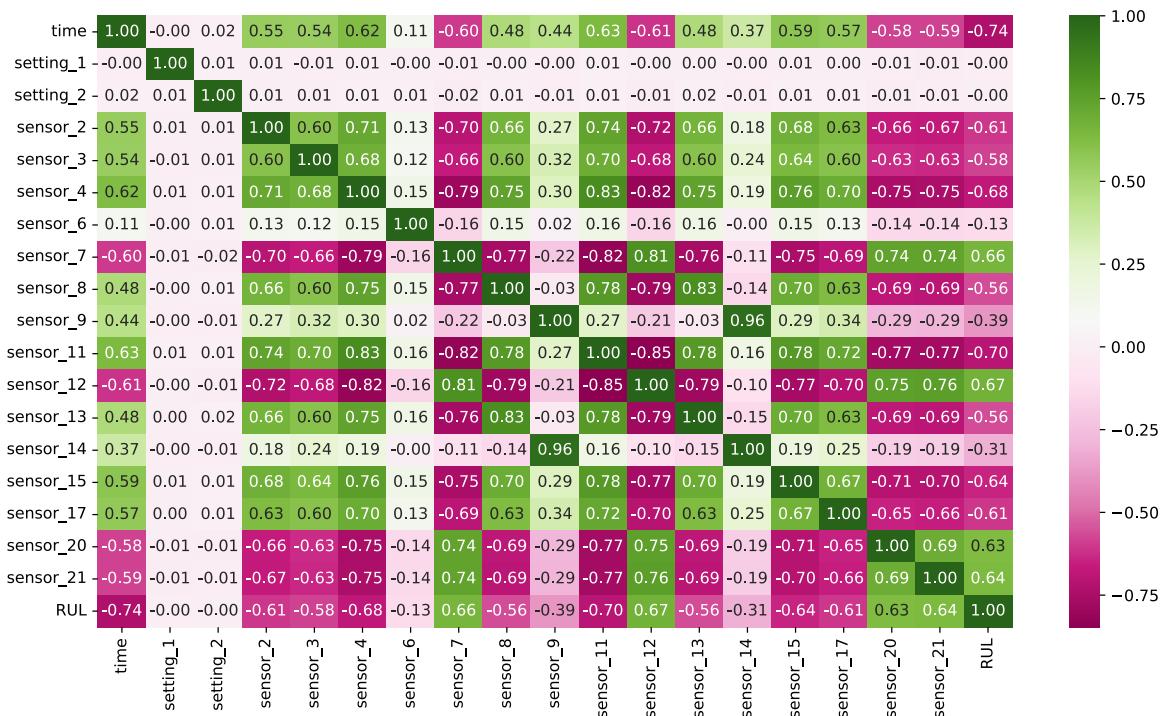
$$r = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}} \quad (20)$$

O Gráfico 4 apresenta o grau de linearidade entre as variáveis presentes nos dados de treinamento, em que os valores próximos a 1 (células verdes) indicam uma relação linearmente proporcional. Por exemplo, os sensores 14 e 9 apresentam uma correlação positiva de 0,96, o que indica que, quando o valor do sensor 14 aumenta, o valor do sensor 9 também aumenta e vice-versa.

Já os valores próximos a -1 (células rosas) indicam uma relação inversamente proporcional. Por exemplo, as *features* RUL e *time* possuem uma correlação negativa de -0,74, o que significa que, à medida que a *feature time* aumenta, a RUL diminui. Essa informação é relevante para entender a relação entre as variáveis e pode ser útil para ajustar o modelo de aprendizado de máquina.

Conforme o processo de modelagem avançou, novas técnicas de pré-processamento foram aplicadas aos dados. Essas técnicas, juntamente com a modelagem e as métricas do modelo treinado, serão apresentadas na próxima seção.

Gráfico 4 - Correlação de Pearson nos dados de treinamento do dataset FD001.



5 MODELAGEM INCREMENTAL

A modelagem de dados envolve o treinamento do modelo utilizando dados históricos e, posteriormente, testando-o com um conjunto de dados diferente daquele utilizado no treinamento, denominado conjunto de teste. Durante essa etapa, são utilizadas métricas de avaliação para determinar qual modelo apresenta o melhor desempenho.

Inicialmente, planejou-se criar modelos utilizando três algoritmos diferentes: *Ridge*, *Lasso* e *Random Forest*. No entanto, durante os experimentos da Etapa 1, um novo modelo, o LightGBM, foi incluído. Para desenvolver o melhor modelo para previsão da RUL, foi empregada a metodologia de modelagem incremental. Essa abordagem envolve a divisão da modelagem em etapas menores, permitindo que cada uma seja desenvolvida, testada e aprimorada antes de prosseguir para a próxima. Em cada etapa, foi implementada uma técnica de pré-processamento diferente, com o objetivo de aprimorar as métricas dos modelos. Ao todo, foram realizadas sete etapas, e cada uma delas envolveu vários experimentos, que são conjuntos de execuções de modelos com diferentes hiperparâmetros ou técnicas de pré-processamento. Os experimentos foram conduzidos seguindo a metodologia descrita a seguir:

1. Pré-processamento: Além do pré-processamento base citado na Seção 4.3, foi usado um específico para cada experimento.
2. Treinamento do modelo: Os quatro modelos foram treinados no mesmo conjunto de dados gerado no pré-processamento.
3. Avaliação dos resultados: As métricas MAE, RMSE e R^2 foram calculadas para o conjunto de teste a fim de avaliar o melhor modelo.

Os modelos deste trabalho foram desenvolvidos utilizando a biblioteca *scikit-learn* em Python, com exceção do LightGBM, que foi criado usando a biblioteca LightGBM. Os experimentos foram criados e gerenciados pela biblioteca MLFlow. Ela permite gerenciar o ciclo de vida do modelo que vai desde a sua criação até o *deploy*. Neste trabalho, só será usada a funcionalidade de rastreamento dos experimentos e comparação de resultados. Com o MLFlow, foi possível filtrar o melhor modelo com base em uma métrica escolhida, além de acessar informações sobre os hiperparâmetros utilizados e visualizar em gráficos a previsão dos modelos.

5.1 Experimentos com *dataset* FD001

Foram realizados uma série de experimentos utilizando apenas os dados do *dataset* FD001. Em seguida, o experimento que obteve os melhores resultados foi aplicado aos demais *datasets*. O nome de cada uma das seções abaixo se refere a etapa realizada.

5.1.1 Etapa 1: Seleção do melhor modelo com AutoML

Ao iniciar o processo de modelagem, surgiu a questão de quais algoritmos seriam mais adequados para o treinamento e teste dos dados do Turbofan. Os algoritmos *Lasso*, *Ridge*, *Random Forest* e LightGBM, mencionados anteriormente, não foram selecionados de forma arbitrária. A escolha dos algoritmos *Lasso*, *Ridge* e *Random Forest* baseou-se em trabalhos relacionados que os recomendavam. O LightGBM foi selecionado posteriormente nesta etapa.

Assim, o objetivo desta etapa é escolher o quarto modelo de aprendizado de máquina mais apropriado para prever a RUL nos dados da base FD001, ao mesmo tempo em que se avaliou os três algoritmos mencionados (*Lasso*, *Ridge* e *Random Forest*) são adequados para o conjunto de dados do Turbofan.

Para simplificar o processo de seleção, foi utilizada a biblioteca Pycaret, uma ferramenta de AutoML para Python que automatiza o treinamento e avaliação de diversos modelos de aprendizado de máquina em poucas linhas de código. O experimento com o Pycaret foi o primeiro a ser executado e tem como objetivo identificar o melhor modelo para prever a RUL. Os dados das 25 features (a coluna *unit number* foi removida) foram usados como entrada do modelo.

A biblioteca Pycaret oferece uma ampla variedade de modelos de regressão, dos quais 29 modelos diferentes foram avaliados neste experimento. A Tabela 3 apresenta os dez modelos que tiveram melhor desempenho em ordem crescente de acordo com a métrica RMSE, além de exibir as métricas MAE e R^2 para o conjunto de teste.

Pode-se observar que o modelo LightGBM apresentou a melhor métrica RMSE, ou seja, esse modelo teve o menor valor de RMSE entre todos os modelos avaliados, cujo resultado foi de 37,77. Além disso, os modelos *Random Forest*, *Lasso* e *Ridge* também se destacaram entre os dez melhores modelos, indicando que a escolha desses modelos foi adequada. Apesar do algoritmo Lasso ocupar a nona posição, ele foi escolhido por ser capaz de selecionar as variáveis. Dessa forma, os algoritmos a serem treinados nas próximas etapas são dois lineares (*Lasso* e *Ridge*) e dois não lineares(*Random Forest* e LightGBM).

Tabela 3 - Resultados do experimento com PyCaret.

Nome do modelo	MAE (ciclos)	RMSE (ciclos)	R ²
<i>Light Gradient Boosting Machine</i>	27,33	37,77	0,69
<i>Gradient Boosting Regressor</i>	27,55	37,85	0,69
<i>Extra Trees Regressor</i>	27,58	37,92	0,68
<i>Random Forest Regressor</i>	27,62	38,05	0,68
<i>Bayesian Ridge</i>	32,12	41,46	0,62
<i>Linear Regression</i>	32,12	41,46	0,62
<i>Ridge Regression</i>	32,12	41,46	0,62
<i>Least Angle Regression</i>	32,12	41,46	0,62
<i>Lasso Regression</i>	32,12	41,49	0,62
<i>K Neighbors Regressor</i>	30,80	42,31	0,61

5.1.2 Etapa 2: Análise dos modelos com parâmetros padrões

Nesta etapa, foram realizados treinamentos dos quatro algoritmos selecionados usando os hiperparâmetros padrões definidos pela biblioteca *scikit-learn*. O objetivo é avaliar o desempenho desses modelos apenas com as configurações padrão. As mesmas *features* utilizadas na etapa anterior foram empregadas no treinamento desses algoritmos.

No entanto, um experimento adicional foi conduzido com o objetivo de avaliar a importância da *feature time* no desempenho dos modelos. A *feature time* apresenta a maior correlação em módulo com a variável RUL, com um valor de 0,74, conforme pode ser visualizado no Gráfico 4. Assim, foram realizados dois experimentos: o primeiro sem a inclusão da *feature time* no modelo e o segundo considerando a inclusão dessa *feature*. Essa abordagem permitiu uma análise mais aprofundada do impacto da *feature time* nos resultados obtidos pelos modelos.

A Tabela 4 apresenta as métricas MAE, RMSE e R² dos modelos sem a *feature time* como entrada. É possível observar que o modelo LightGBM apresentou o menor valor de RMSE, com 46,07. Apesar disso, essa métrica pode ser melhorada pois esse resultado ainda é inferior ao modelo do LightGBM da Etapa 1, que obteve um RMSE menor, cujo valor foi de 37,77. E também é inferior ao pior RMSE entre os trabalhos relacionados citados no Capítulo 3 cujo valor é 29,73.

É importante ressaltar que as métricas de erro, como o MAE e o RMSE, fornecem uma medida da magnitude dos erros do modelo. Por exemplo, se o MAE esperado é de 37,24 ciclos, isso significa que o modelo pode ter uma margem de erro $\pm 37,24$ em relação à previsão da RUL.

Em outras palavras, o MAE de 37,24 indica que, em média, o modelo pode apresentar um desvio de 37,24 ciclos em suas previsões da RUL. Por exemplo, se o modelo

prevê a RUL igual a 100 ciclos, espera-se que o valor real possa estar em um intervalo de 62,76 a 137,24 ciclos, considerando a margem de erro permitida pelo MAE.

Tabela 4 - Métricas para o conjunto de teste dos experimentos da Etapa 2 sem a *feature time*.

Nome do modelo	MAE (ciclos)	RMSE (ciclos)	R^2
<i>Lasso</i>	37,24	48,50	0,324
<i>Ridge</i>	37,19	48,42	0,326
<i>Random Forest</i>	35,06	46,39	0,383
LightGBM	34,59	46,07	0,390

Após realizar os experimentos sem a variável *time*, conduziu-se um novo experimento incluindo-a, cujos resultados são apresentados na Tabela 5. Nessa tabela, nota-se que o modelo LightGBM manteve a menor métrica RMSE, igual a 41,30. Dessa forma, ao incluir a *feature time* para o modelo LightGBM obteve-se uma melhoria de 10,38%. Essa melhoria foi calculada pela diferença entre as métricas (antiga e nova) dividida pela métrica antiga.

A melhoria ao incluir a *time* era esperada em virtude da forte correlação negativa entre a RUL e essa variável, conforme evidenciado no Gráfico 4. Cabe ressaltar que, tanto na Tabela 4 quanto na Tabela 5, os modelos baseados em árvores (*Random Forest* e LightGBM) apresentaram os menores valores de RMSE em relação aos modelos lineares (*Lasso* e *Ridge*). Esse comportamento é esperado, uma vez que a degradação do motor Turbofan não é linear (PENG et al., 2021). Dessa forma, os modelos baseados em árvores são capazes de se ajustar melhor aos dados e, consequentemente, apresentam erros menores.

Tabela 5 - Métricas para o conjunto de teste dos experimentos da Etapa 2 com a *feature time*.

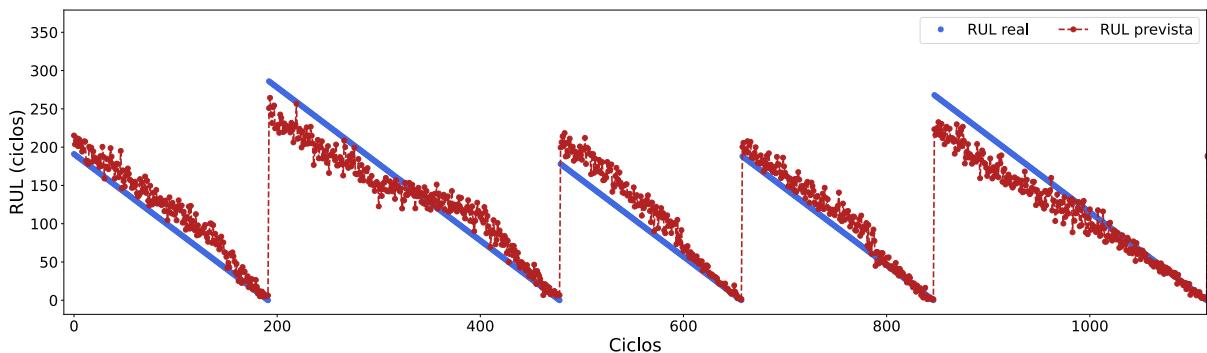
Nome do modelo	MAE (ciclos)	RMSE (ciclos)	R^2
<i>Lasso</i>	33,36	42,97	0,457
<i>Ridge</i>	33,12	42,93	0,470
<i>Random Forest</i>	31,62	41,45	0,506
LightGBM	31,30	41,30	0,510

Como o modelo LightGBM apresentou a melhor métrica (menor RMSE), foi plotado no Gráfico 5, que possibilita a comparação entre as previsões (RUL prevista) e os dados reais (RUL real) dos primeiros 1500 ciclos do conjunto de dados. O modelo utilizado foi o LightGBM treinado na Etapa 2 (LightGBM-etapa-2), considerando a inclusão da *feature time*. No Gráfico 5a, que se refere aos dados de treinamento, a previsão do modelo

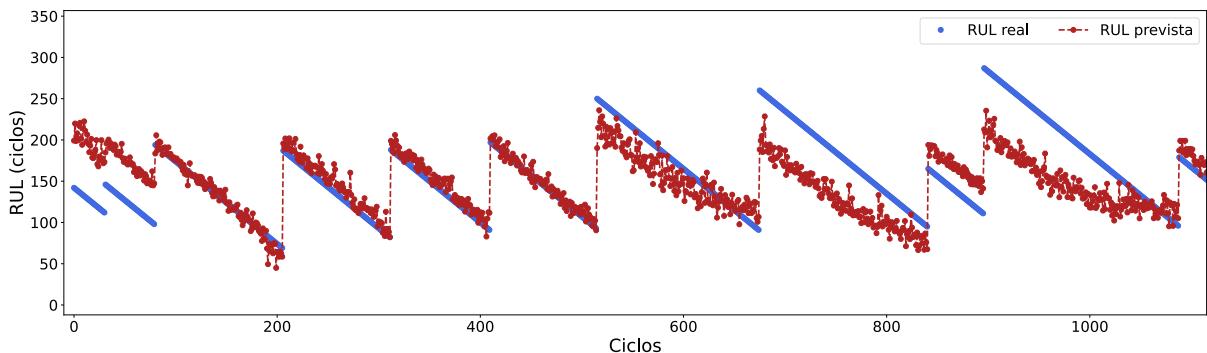
ficou mais próxima aos dados reais. Já nos dados de teste, exibidos pelo Gráfico 5b, a previsão apresentou maior afastamento dos dados reais nas duas primeiras e nas últimas campanhas. Nessas campanhas, há um *bias*, ou seja, um erro sistemático na previsão do modelo. Isso indica que o modelo foi capaz de se ajustar bem aos dados de treino, mas não conseguiu generalizar tão bem para novos dados. Apesar disso, o modelo manteve a tendência de decaimento da RUL na previsão.

Conforme mencionado no final da Seção 4.1, no conjunto de teste não há o decaimento da RUL até zero, como pode ser visto no Gráfico 5b. Essa característica já está presente nos dados brutos, uma vez que esses dados são utilizados em competições de aprendizado de máquina. Essa diferença, por sua vez, torna a criação desse tipo de modelo mais desafiadora.

Gráfico 5 - Comparação entre os valores previstos pelo modelo LightGBM-etapa-2 e os valores reais da RUL no *dataset* FD001.



(a) Valores previstos *versus* real para o conjunto de treino.

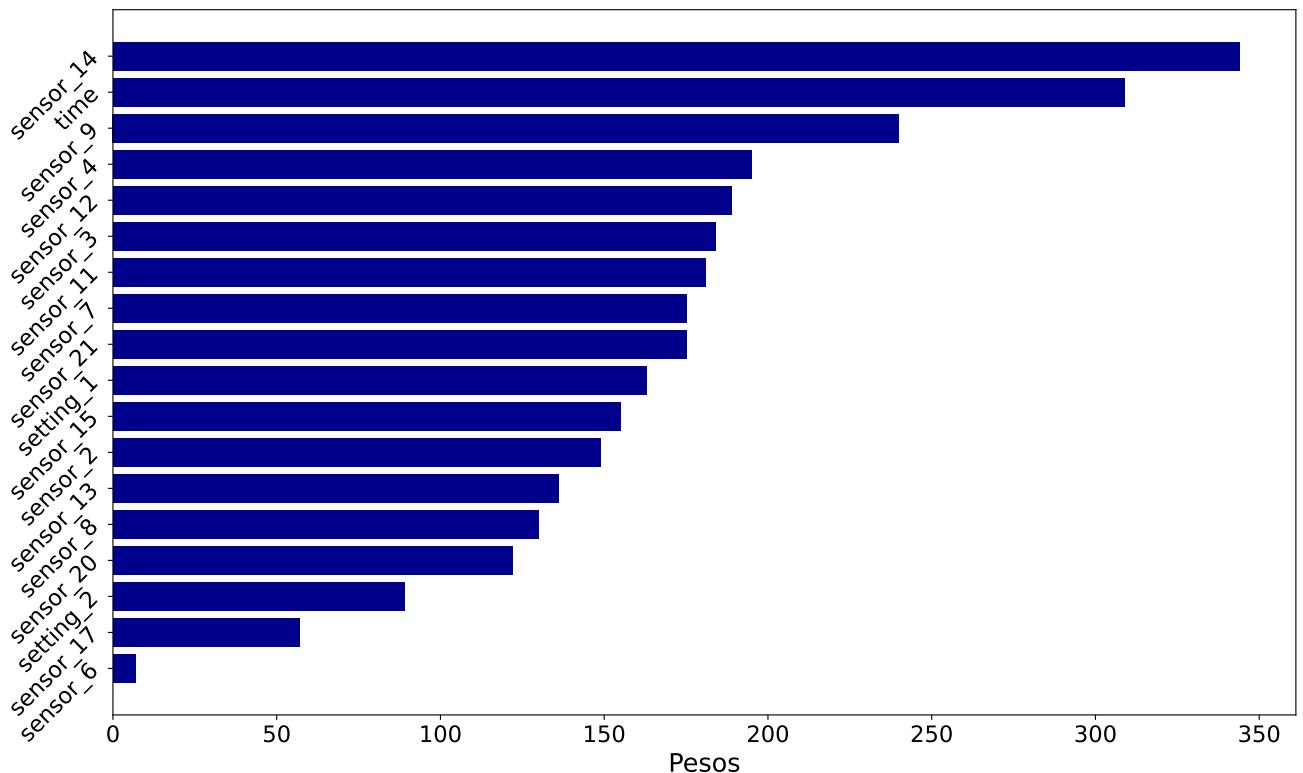


(b) Valores previstos *versus* real para o conjunto de teste.

Além do Gráfico 5, que compara os valores previstos e reais, também foi criado o Gráfico 6, que exibe a importância das variáveis do modelo LightGBM-etapa-2. Nota-se que a variável *sensor_1* não está presente, uma vez que é uma variável *flat*, ou seja, ela

não varia ao longo do tempo. Além disso, as *features* mais relevantes são *sensor_14* e *time*, apresentando os maiores pesos para o modelo. Isso significa que essas variáveis têm maior correlação com a RUL para o modelo LightGBM-etapa-2.

Gráfico 6 - Importância das variáveis para o modelo LightGBM-etapa-2 no dataset FD001.



Com base nos resultados obtidos, a decisão foi tomada de incluir a *feature time* no treinamento dos modelos nas etapas seguintes, bem como ajustar os hiperparâmetros dos modelos.

5.1.3 Etapa 3: Melhoria dos modelos baseados em árvore

O objetivo desta etapa é aprimorar o desempenho dos modelos baseados em árvore, buscando os melhores hiperparâmetros. Ao contrário dos modelos lineares (*Lasso* e *Ridge*), que possuem apenas o hiperparâmetro alfa para ajustar, os modelos em árvore apresentam vários hiperparâmetros, o que dificulta a busca pela melhor combinação de forma empírica. Assim, utilizou-se o Optuna, uma biblioteca que busca os melhores hiperparâmetros que emprega o teorema de Bayes para encontrá-los, a fim de melhorar as métricas dos modelos em árvore. Vale destacar que, nos experimentos com os algoritmos lineares, não houve

diferença significativa nas métricas ao utilizar os hiperparâmetros padrões da biblioteca em comparação com os hiperparâmetros definidos empiricamente.

Para o modelo *Random Forest*, foram buscados os seguintes hiperparâmetros: *max_depth*, *max_features*, *n_estimators* e *min_samples_split*. Já para o modelo LightGBM, os hiperparâmetros buscados foram: *max_depth*, *n_estimators* e *boosting_type*. A Tabela 6 mostra os valores dos hiperparâmetros encontrados pelo Optuna.

Tabela 6 - Hiperparâmetros encontrados pelo Optuna.

Hiperparâmetros	Random Forest	LightGBM
<i>max_depth</i>	8	4
<i>n_estimators</i>	186	36
<i>max_features</i>	5	N/A ^a
<i>min_samples_split</i>	6	N/A
<i>boosting_type</i>	N/A	GBDT

^a Não se aplica.

A Tabela 7 apresenta as métricas obtidas no conjunto de teste para os modelos otimizados, enquanto a Tabela 8 exibe as melhorias percentuais alcançadas em relação aos experimentos realizados na Etapa 2, na qual não foram utilizados hiperparâmetros otimizados. O cálculo da melhoria foi realizado levando em consideração as métricas anteriores, que estão apresentadas na Tabela 5, e as métricas atualizadas, que estão na Tabela 7. Conforme os resultados, os melhores hiperparâmetros contribuiram para melhorias de até 3,62% no R² dos modelos em árvore. Além disso, é possível observar que o *Random Forest* obteve o menor RMSE (40,64) em comparação ao LightGBM (40,70). Dessa forma, ao contrário do que ocorria anteriormente, o *Random Forest* passou a apresentar um desempenho melhor em relação ao LightGBM.

Tabela 7 - Métricas para o conjunto de teste dos experimentos da Etapa 3 com os melhores hiperparâmetros nos modelos baseados em árvore.

Nome do modelo	MAE (ciclos)	RMSE (ciclos)	R ²
<i>Random Forest</i>	30,48	40,64	0,525
LightGBM	30,61	40,70	0,524

Com base na melhoria das métricas observadas na Tabela 8, a utilização do Optuna foi adotada para aprimorar o desempenho dos modelos baseados em árvore.

Tabela 8 - Melhoria percentual dos modelos baseados em árvore otimizados com Optuna em relação aos modelos baseados em árvore da Etapa 2.

Nome do modelo	MAE (%)	RMSE (%)	R ² (%)
Random Forest	3,61	1,95	3,62
LightGBM	2,20	1,45	2,67

5.1.4 Etapa 4: Transferência dos dados de treino para os dados de teste

Observa-se, pelo Gráfico 7b, que o conjunto de teste não apresenta informações sobre o momento da falha, ou seja, quando a RUL atinge o valor zero, diferentemente do conjunto de treinamento, exibido no Gráfico 7a.

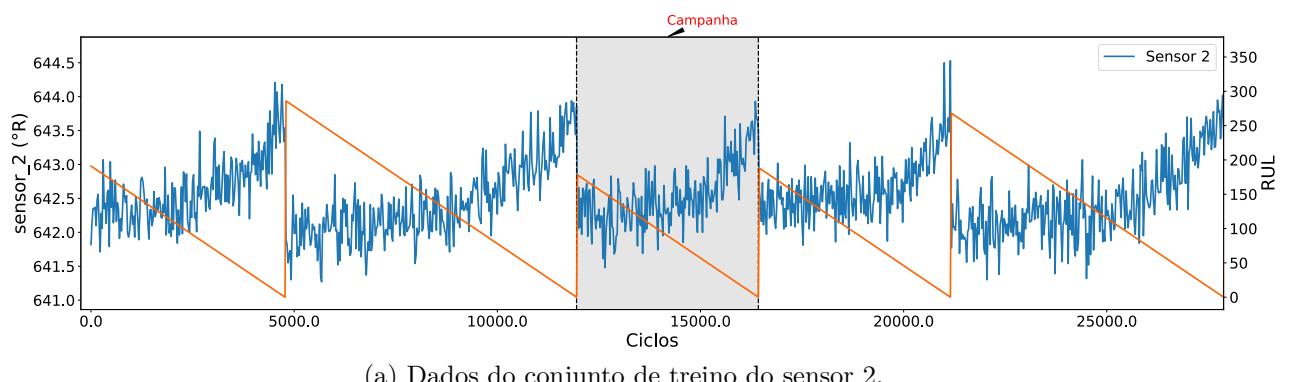
Devido a essa diferença nos dados, os dois conjuntos não são exatamente iguais. Os dados de treinamento abrangem todo o período da campanha, incluindo o final, quando os sensores apresentam uma taxa de variação (derivada) mais alta. Em contrapartida, os dados de teste abrangem apenas a parte inicial da campanha, com pouca variação, como evidenciado no Gráfico 7b, onde o sensor 2 não exibe os picos presentes no Gráfico 7a. Essa baixa variação do sensor 2 nos dados de teste resulta em informações limitadas para o teste do modelo.

Portanto, o objetivo desta etapa é transferir 4% das campanhas do conjunto de treinamento para o conjunto de teste, com o intuito de avaliar e aprimorar as métricas dos quatro modelos analisados. O conjunto de treinamento consiste em 100 campanhas, o que implica em transferir 4 campanhas para o conjunto de teste. Cada campanha tem aproximadamente 250 amostras.

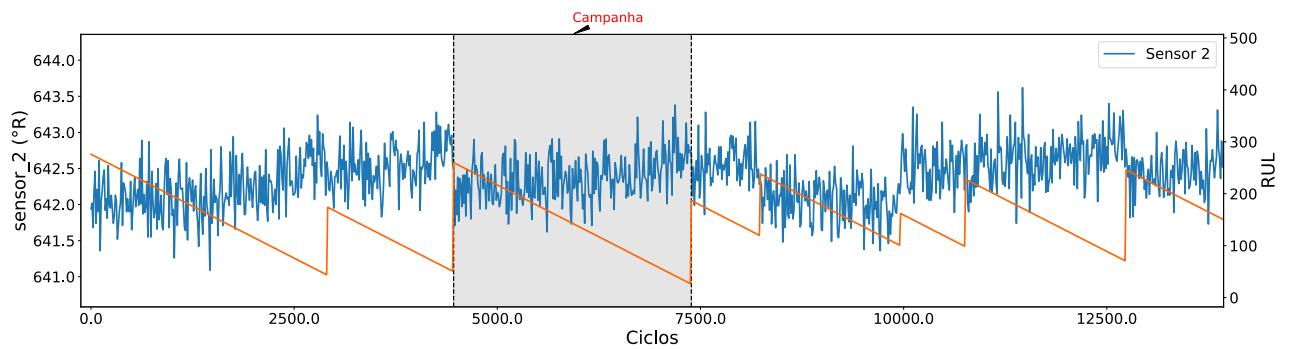
A Tabela 9 apresenta as métricas obtidas no experimento com a transferência de dados, enquanto a Tabela 10 exibe as melhorias percentuais alcançadas em relação aos modelos da Etapa 2, que não foi empregada essa técnica de pré-processamento. Vale ressaltar que essa comparação foi realizada de forma subjetiva, pois as métricas da etapa 2 não são diretamente comparáveis às métricas da etapa 4. Isso ocorre porque, na etapa 4, houve uma alteração nos dados de treinamento e teste, ou seja, as métricas não estão sendo calculadas com a mesma base de dados. Portanto, não houve alteração no modelo em si, apenas nos dados fornecidos a ele. Como será detalhado no próximo parágrafo, o objetivo é avaliar como a variação dos dados dos sensores no final da campanha contribui para a melhora na métrica do modelo.

O cálculo da melhoria foi realizado levando em consideração as métricas anteriores, que estão apresentadas na Tabela 5, e as métricas atualizadas, que estão na Tabela 9. Na Tabela 10, é possível notar uma melhoria, especialmente no R², que chegou a 7,49% no modelo *Lasso*. Isso evidencia que os dados com maior variação fornecem mais informa-

Gráfico 7 - Comparação entre o sensor 2 do conjunto de treino e do conjunto de teste.



(a) Dados do conjunto de treino do sensor 2.



(b) Dados do conjunto de teste do sensor 2.

ções para o modelo do que aqueles com pouca variação, como os coletados no início da campanha. Esse comportamento dos dados pode ser observado no Gráfico 1b, onde é perceptível um crescimento acentuado do *sensor_2* no final de cada campanha. Além disso, o modelo LightGBM obteve o menor RMSE em comparação aos outros modelos, o que também foi observado nas Etapas 1 e 2.

Tabela 9 - Métricas para o conjunto de teste dos experimentos da Etapa 4 com transferência dos dados de treino para os dados de teste.

Nome do modelo	MAE (ciclos)	RMSE (ciclos)	R ²
<i>Lasso</i>	33,02	42,53	0,494
<i>Ridge</i>	33,04	42,05	0,494
<i>Random Forest</i>	30,39	40,25	0,546
LightGBM	30,36	40,26	0,546

Tabela 10 - Melhoria percentual dos modelos da Etapa 4 em relação aos modelos da Etapa 2.

Nome do modelo	MAE (%)	RMSE (%)	R ² (%)
<i>Lasso</i>	1,02	1,02	7,49
<i>Ridge</i>	0,24	2,05	4,86
<i>Random Forest</i>	0,30	0,96	3,85
LightGBM	0,82	1,08	4,03

Devido à melhoria observada em comparação subjetiva aos modelos da Etapa 2, conforme pode ser visto na Tabela 10, a técnica de transferência dos dados de treino para o conjunto de dados de teste foi adotada nos experimentos seguintes.

5.1.5 Etapa 5: Atenuação dos ruídos nos dados de entrada

Essa etapa tem como objetivo aplicar técnicas de suavização de sinal para reduzir o ruído presente nos dados. Como foi observado no Gráfico 2a, os dados de entrada exibem um nível de ruído que pode afetar a precisão do modelo. Foram utilizadas duas técnicas: a média móvel e o filtro SG. Foi definida uma janela de 12 ciclos tanto para a média móvel quanto para o filtro SG, e optou-se por utilizar um polinômio de grau dois para o cálculo do filtro SG. Foram feitos alguns experimentos empiricamente para se chegar no valor da janela e do polinômio.

As Tabelas 11 e 12 apresentam as métricas do modelo após a aplicação da média móvel e do filtro SG, respectivamente. Observa-se que em ambas as tabelas, o modelo

LightGBM obteve o menor RMSE em comparação aos demais modelos dessa etapa. Além disso, é importante destacar que o filtro SG apresentou melhores métricas do que a média móvel, devido ao fato de que a média móvel produz dados nulos em cada janela de tamanho definido, reduzindo assim a quantidade de dados disponíveis para treinamento e teste do modelo². Esse fato pode ser comprovado pela métrica do modelo LightGBM com média móvel com RMSE igual a 41,26 e com filtro SG com RMSE menor igual a 40,61.

Tabela 11 - Métricas para o conjunto de teste dos experimentos da Etapa 5 com média móvel.

Nome do modelo	MAE (ciclos)	RMSE (ciclos)	R ²
<i>Lasso</i>	32,94	42,47	0,461
<i>Ridge</i>	32,94	42,54	0,463
<i>Random Forest</i>	31,24	41,51	0,487
LightGBM	31,23	41,26	0,493

Tabela 12 - Métricas para o conjunto de teste dos experimentos da Etapa 5 com SG.

Nome do modelo	MAE (ciclos)	RMSE (ciclos)	R ²
<i>Lasso</i>	33,05	42,44	0,496
<i>Ridge</i>	33,10	42,42	0,496
<i>Random Forest</i>	30,95	40,87	0,532
LightGBM	30,52	40,61	0,538

As Tabelas 13 e 14 apresentam a comparação percentual dos modelos dessa etapa com os modelos da Etapa 4. Para calcular essas porcentagens, comparou-se as métricas antigas que se encontram na Tabela 9 com as novas métricas das Tabelas 11 e 12. Observa-se que a aplicação da técnica de suavização de sinal resultou em uma piora nas métricas dos modelos, indicada pelas porcentagens negativas apresentadas nas tabelas. Isso sugere que os ruídos presentes nos dados dos sensores contêm informações importantes para o modelo.

Como não houve melhorias significativas nesses experimentos, a suavização dos dados não foi utilizada nas próximas etapas.

² Esses dados nulos surgem porque o cálculo da média móvel requer os últimos 12 dados. Como nas primeiras 12 amostras não há dados anteriores, a biblioteca Pandas insere automaticamente valores nulos.

Tabela 13 - Comparação percentual das métricas dos modelos da Etapa 5 com média móvel em relação aos modelos da Etapa 4

Nome do modelo	MAE (%)	RMSE (%)	R ² (%)
<i>Lasso</i>	0,24	0,14	-7,16
<i>Ridge</i>	0,36	-1,17	-6,70
<i>Random Forest</i>	-2,80	-3,12	-12,11
LightGBM	-2,87	-2,48	-10,75

Tabela 14 - Comparação percentual das métricas dos modelos da Etapa 5 com filtro SG em relação aos modelos da Etapa 4.

Nome do modelo	MAE (%)	RMSE (%)	R ² (%)
<i>Lasso</i>	-0,09	0,21	0,40
<i>Ridge</i>	-0,18	-0,88	0,40
<i>Random Forest</i>	-1,84	-1,54	-2,63
LightGBM	-0,53	-0,87	-1,49

5.1.6 Etapa 6: Adição de features polinomiais

O objetivo dessa nova etapa é melhorar as métricas dos modelos, principalmente os lineares, por meio da adição de variáveis polinomiais aos dados de entrada. Como explicado na Subseção 2.1.3, as variáveis polinomiais consistem na multiplicação das variáveis de entrada entre si, introduzindo características não lineares no modelo. Para realizar essa operação, utilizou-se o método *Polynomial Features* da biblioteca *scikit-learn*, que gera novas *features* a partir da combinação polinomial das variáveis originais. Optou-se pelo grau dois nesse método. Portanto, espera-se que os modelos lineares melhorem suas métricas.

Ao utilizar o PF nas 25 *features*, foram geradas um total de 350 *features*. No entanto, utilizar todas essas *features* como entrada tornaria o treinamento do algoritmo lento e aumentaria a complexidade do modelo devido ao aumento da dimensionalidade do problema. Por essa razão, optou-se por utilizar um algoritmo genético para selecionar as melhores variáveis geradas e reduzir a dimensão do modelo. O algoritmo genético foi implementado utilizando a linguagem Python, por meio de um código desenvolvido pelos autores (GRANATYR; PACHOLOK, 2022).

Primeiramente, foi definida a função de aptidão (*fitness function*) do algoritmo genético, a qual se baseou em um modelo de regressão linear tradicional. O desempenho desse modelo foi avaliado utilizando a métrica do RMSE. Em seguida, foi estabelecido o tamanho da população, composta por 150 indivíduos, em que cada um representa uma combinação específica de características do conjunto de dados.

A seleção dos pais foi realizada através do método de torneio (*tournament selection*). Nesse processo, 80 indivíduos foram selecionados aleatoriamente da população e competiram entre si. Os dois melhores indivíduos de cada competição foram escolhidos como pais para a próxima geração.

Durante a reprodução, aplicou-se a recombinação dos pais selecionados, combinando suas características para gerar descendentes que herdaram características promissoras. Além disso, foi aplicada uma taxa de mutação de 0.05 durante a reprodução, introduzindo pequenas alterações aleatórias nas características dos indivíduos descendentes, possibilitando explorar novas soluções no espaço de busca.

O algoritmo genético foi executado por um mínimo de 100 gerações e um máximo de 500 gerações. Durante a execução, monitorou-se o progresso do algoritmo, registrando os percentis (100, 75, 50, 25, 0) do RMSE em cada geração para fornecer informações sobre o desempenho ao longo do tempo. Além disso, acompanhou-se a idade média dos indivíduos na população a cada geração, permitindo analisar a evolução do algoritmo. Por fim, os melhores indivíduos obtidos foram avaliados no conjunto de teste.

A Tabela 15 apresenta as métricas dos experimentos dessa etapa, enquanto a Tabela 16 exibe as melhorias percentuais alcançadas em relação aos modelos da Etapa 4. Para calcular a melhoria, considerou-se as métricas antigas na Tabela 9 e as novas na Tabela 15. Pode-se observar que o modelo LightGBM novamente apresentou o menor RMSE, igual a 40,20. Além disso, apenas o modelo *Ridge* apresentou uma melhoria significativa nas métricas em comparação ao mesmo modelo da Etapa 4. O desempenho do modelo *Ridge* (RMSE igual a 40,25) equiparou-se ao modelo *Random Forest* da Etapa 4, enquanto o modelo *Lasso* apresentou uma piora na métrica, conforme as porcentagens negativas na Tabela 16. A melhoria do *Ridge* pode ser atribuída à inclusão de variáveis polinomiais no modelo, que introduziu a capacidade do modelo em capturar dependências não lineares entre as variáveis, o que pode ser crucial nesses casos em que as relações entre as variáveis não são estritamente lineares. Já a piora do *Lasso* ocorreu porque o algoritmo genético foi utilizado para seleção de variáveis, o que não foi benéfico, uma vez que o *Lasso* já possui um algoritmo integrado para essa finalidade. Assim, o modelo *Lasso* acabou usando dois algoritmos de seleção de variáveis, o que prejudicou seu desempenho.

Tabela 15 - Métricas para o conjunto de teste dos experimentos da Etapa 6.

Nome do modelo	MAE (ciclos)	RMSE (ciclos)	R^2
<i>Lasso</i>	33,21	42,99	0,483
<i>Ridge</i>	30,87	40,25	0,546
<i>Random Forest</i>	30,80	40,56	0,539
LightGBM	30,24	40,20	0,548

Tabela 16 - Comparaçāo percentual das métricas dos modelos da Etapa 6 com PF em relação aos modelos da Etapa 4.

Nome do modelo	MAE (%)	RMSE (%)	R ² (%)
<i>Lasso</i>	-0,58	-1,08	-2,28
<i>Ridge</i>	6,57	4,28	9,52
<i>Random Forest</i>	-1,35	-0,77	-1,30
LightGBM	0,40	0,15	0,36

A partir dos resultados dessa etapa, o pré-processamento com PF foi utilizado somente no modelo *Ridge*.

5.1.7 Etapa 7: Treinamento e teste na região de interesse

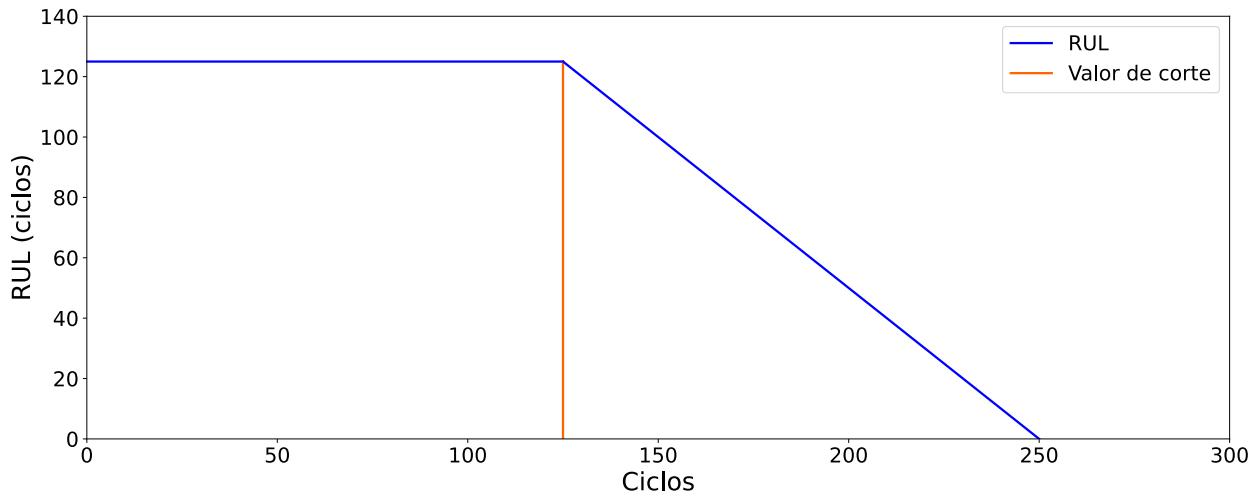
Após constatar que a técnica de PF apresentou melhorias apenas no modelo *Ridge*, foram exploradas outras alternativas para aprimorar o desempenho dos modelos. Uma dessas alternativas consistiu em utilizar apenas os dados da região de interesse (*Region of Interest* - ROI) durante o treinamento e teste dos modelos. Especificamente, os modelos foram treinados considerando a RUL constante acima de 125 ciclos, e avaliados apenas com a RUL abaixo desse valor. Essa região é considerada a mais crítica para a manutenção do Turbofan, pois está próxima do fim de sua vida útil. A escolha do valor de 125 ciclos foi baseada no trabalho de Peng *et al.* (PENG et al., 2021). O objetivo dessa abordagem é utilizar somente os dados mais relevantes para o problema, os quais estão localizados na ROI.

No Gráfico 8, é possível observar uma reta vertical laranja que marca o ponto a partir do qual os modelos foram avaliados. Os dados da RUL maiores que 125 ciclos (valor de corte) são considerados constantes, uma vez que, de acordo com (PENG et al., 2021), a probabilidade de ocorrer uma falha no início da campanha é muito pequena. No entanto, abaixo dos 125 ciclos, a probabilidade de falha aumenta, pois o motor começa a apresentar uma degradação mais acentuada, o que pode levar à ocorrência de falhas.

A Tabela 17 apresenta as métricas obtidas pelos modelos dessa etapa, enquanto que a Tabela 18 apresenta a melhoria alcançada em relação aos experimentos da Etapa 4. É importante salientar que o modelo *Ridge* está com pré-processamento PF da Etapa 6, já os demais modelos não estão. Pode-se observar na Tabela 17 que o modelo LightGBM novamente apresentou o menor RMSE, igual a 19,16.

Na Tabela 18, é perceptível que a utilização dos dados da ROI resultou em uma melhoria de 50% no RMSE de todos os modelos. Esse efeito se deve ao fato de que os dados coletados no final da campanha de um equipamento apresentam uma taxa de

Gráfico 8 - Valor da RUL considerada relevante para o treinamento dos modelos.



variação maior do que os dados coletados no início, conforme mencionado na Etapa 4 e evidenciado pelo Gráfico 1b, onde é perceptível um crescimento acentuado do *sensor_2* no final de cada campanha.

Tabela 17 - Métricas para o conjunto de teste dos experimentos da Etapa 7.

Nome do modelo	MAE (ciclos)	RMSE (ciclos)	R ²
<i>Lasso</i>	17,64	21,68	0,519
<i>Ridge^a</i>	16,06	20,44	0,573
<i>Random Forest</i>	15,01	19,67	0,604
<i>LightGBM</i>	14,66	19,16	0,625

^a O pré-processamento do *Ridge* inclui o método PF.

Como o modelo LightGBM apresentou a melhor métrica, ou seja, o menor valor de RMSE foi plotado o Gráfico 9 para comparação entre as previsões feitas pelo modelo LightGBM-etapa-7 (RUL prevista) com os dados reais (RUL real) do ciclo 2750 ao 5000 do conjunto de dados. Ao analisar o Gráfico 9a, que se refere aos dados de treinamento, é possível observar que a previsão do modelo ficou mais próxima dos dados reais. Por outro lado, ao examinar os dados de teste no Gráfico 9b, é possível identificar um *bias* em algumas campanhas, como evidenciado na campanha do ciclo 4250. Essa discrepância ocorre devido à tendência do modelo apresentar métricas inferiores no conjunto de teste em comparação ao conjunto de treinamento. No entanto, de maneira geral, o modelo apresentou um bom desempenho prevendo corretamente o final da vida útil do motor.

Tabela 18 - Comparaçāo percentual das métricas dos modelos da Etapa 7 com os dados na ROI em relação aos modelos da Etapa 4.

Nome do modelo	MAE (%)	RMSE (%)	R ² (%)
<i>Lasso</i>	46,58	49,02	4,82
<i>Ridge</i>	51,39	51,39	13,79
<i>Random Forest</i>	50,61	51,13	9,60
LightGBM	51,71	52,41	12,64

O Gráfico 9c apresenta a importância das variáveis para o modelo LightGBM-etapa-7. Ao analisar o gráfico, pode-se observar que as *features time* e *sensor_14* apresentaram o maior peso, semelhante ao que foi observado no Gráfico 6 da Etapa 2. Entretanto, é importante destacar que, neste gráfico, o peso da variável *time* é significativamente maior em comparação com as outras variáveis. Isso se deve ao fato de que a RUL está diretamente relacionada ao tempo de operação do motor, ou seja, quanto maior o tempo de operação, menor será a vida útil restante do motor. Assim, o modelo LightGBM-etapa-7 considera a variável *time* como um fator crítico para a previsão da RUL. Além disso, a variável *sensor_14* também apresentou um peso considerável, o que indica a importância desse sensor na detecção de falhas do motor.

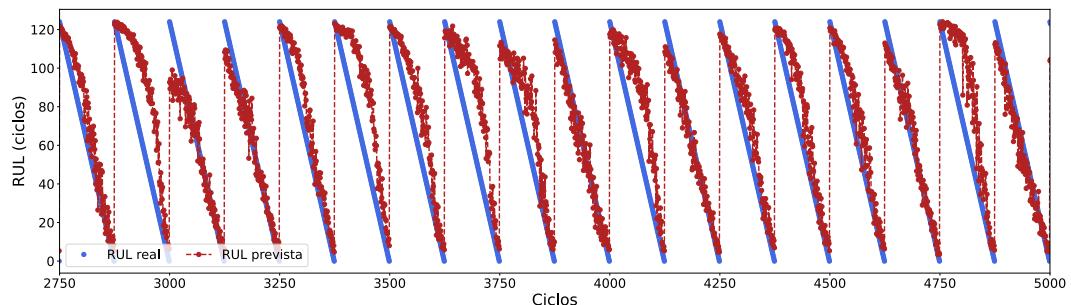
Os resultados da Tabela 18 demonstram que a inclusão dos dados da ROI resultou em uma melhoria significativa nas métricas dos modelos. Dentre as sete etapas executadas, o modelo LightGBM-etapa-7 apresentou o menor RMSE (19,16), o que indica que suas previsões foram mais precisas e próximas dos valores observados. Portanto, o modelo LightGBM-etapa-7 foi selecionado para o experimento com os demais *datasets*, cujo desenvolvimento será detalhado na próxima seção.

5.2 Experimento com outros datasets

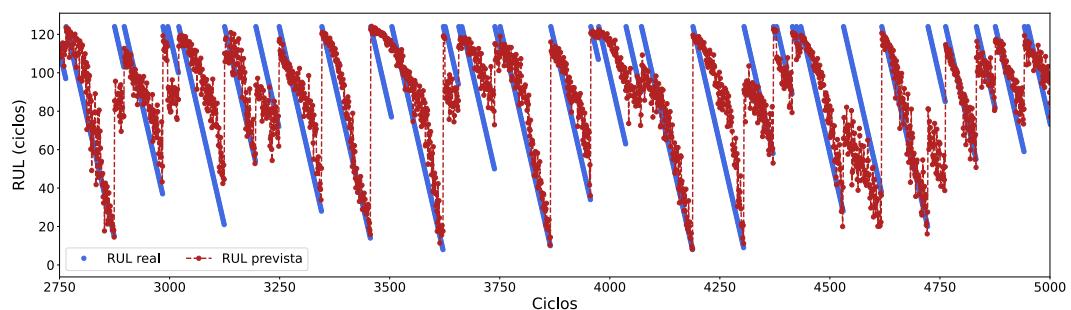
Os experimentos da Etapa 7 resultou no melhor modelo para a previsão da RUL do motor Turbofan, o qual utiliza o algoritmo LightGBM em conjunto com a seguinte sequência de pré-processamento:

1. Normalização padrão dos dados.
2. Seleção de 4% do conjunto de treinamento para compor o conjunto de teste.
3. Busca dos melhores hiperparâmetros por meio da ferramenta Optuna.
4. Utilização dos dados contidos na ROI.

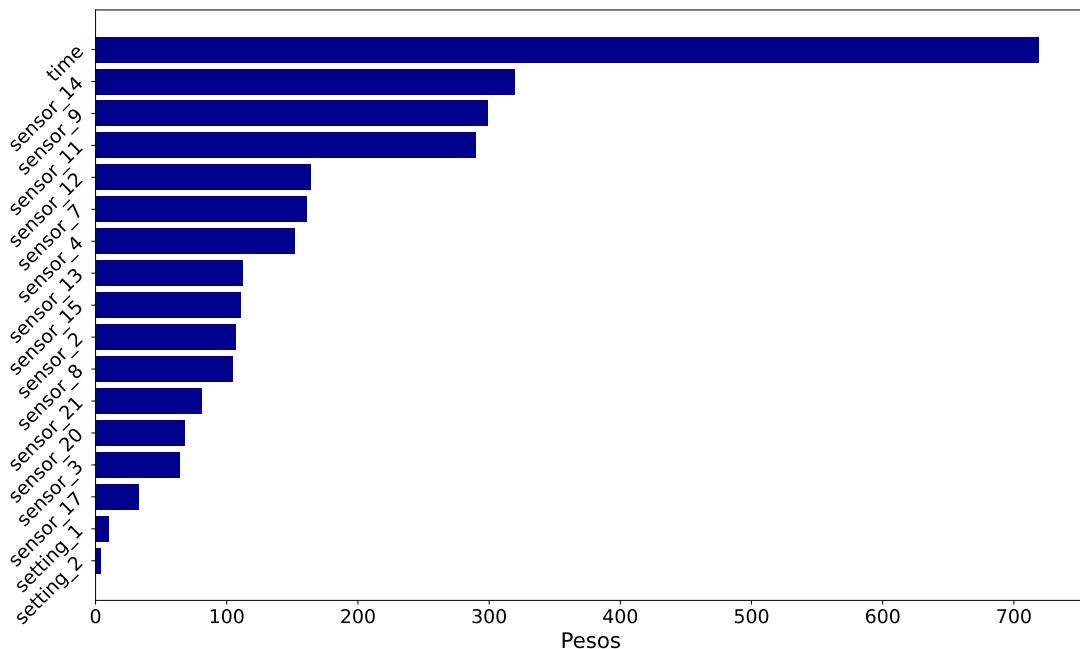
Gráfico 9 - Comparação entre os valores previstos pelo modelo LightGBM-etapa-7 e os valores reais da RUL no *dataset* FD001.



(a) Valores previstos *versus* real para o conjunto de treino.



(b) Valores previstos *versus* real para o conjunto de teste.



(c) Importância das variáveis.

Em virtude dos resultados promissores, o modelo mencionado e as técnicas de pré-processamento foram utilizados para treinar os demais *datasets* (FD002, FD003 e FD004). É importante destacar que esses *datasets* se diferenciam pelo desgaste inicial, conforme descrito na Seção 4.1. A tabela 19 exibe as métricas obtidas para cada um desses *datasets* após o treinamento e teste do modelo LightGBM-etape-7. Observa-se que os resultados de RMSE são semelhantes ao *dataset* FD001, especialmente no *dataset* FD003, o que era esperado, uma vez que ele inicia a campanha com o mesmo desgaste do FD001.

Tabela 19 - Métricas do conjunto de teste para o modelo
LightGBM-etape-7 para os *datasets* FD002 à FD004.

Dataset	MAE (ciclos)	RMSE (ciclos)	R²
FD002	15,78	20,34	0,571
FD003	14,85	19,26	0,593
FD004	16,68	21,40	0,518

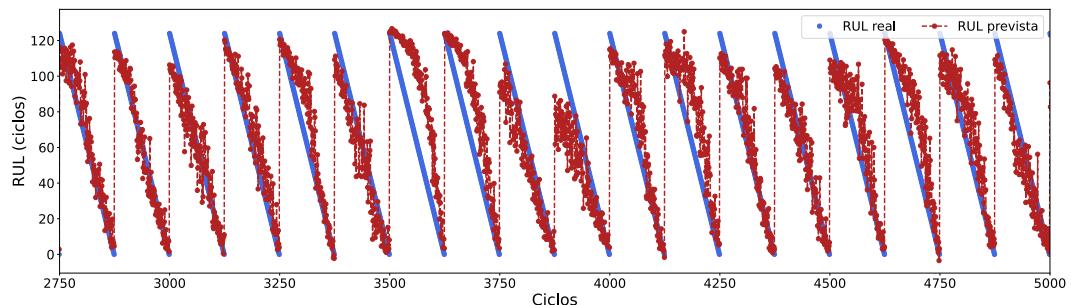
Os Gráficos 10, 11 e 12 permitem comparar os valores previstos pelos modelos aos valores reais para os *datasets* FD002, FD003 e FD004, respectivamente. Além disso, eles mostram a importância de cada variável para cada modelo. Observa-se que, em todos os casos, os valores previstos apresentam maior proximidade com os valores reais no conjunto de treinamento, conforme os Gráficos 10a, 11a e 12a. Entretanto, os valores previstos também apresentam boa aproximação em relação aos valores reais no conjunto de teste, embora em menor grau como pode ser visto nos Gráficos 10b, 11b e 12b. Vale ressaltar que em todos os gráficos de importância das variáveis, a *feature time* foi a mais relevante.

5.3 Discussão dos resultados

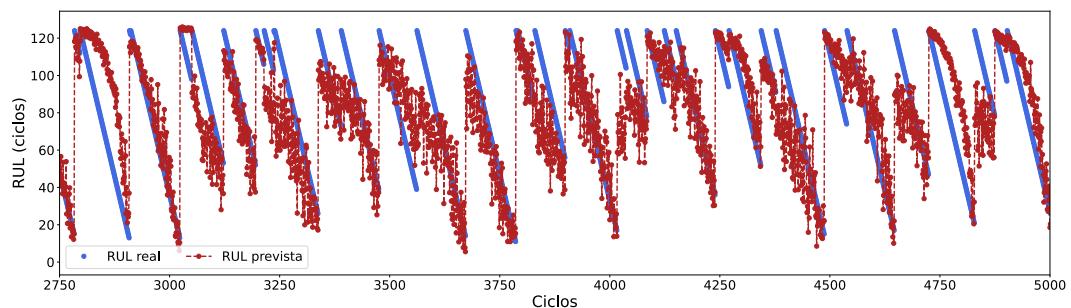
Nas últimas seções buscou-se o melhor modelo para fazer a previsão da RUL do Turbofan, ou seja, aquele modelo que apresenta as melhores métricas. Dessa forma, as métricas MAE e RMSE devem ser próximos de zero, e o R² deve ser próximo de um. Esses resultados são difíceis de alcançar em *datasets* obtidos em situações reais. Por esse motivo foram feitos os experimentos, nas sete etapas anteriores, para alcançar o melhor resultado. O Gráfico 13 resume os resultados da métrica RMSE dos modelos para cada experimento no *dataset* FD001. Optou-se por usar o RMSE na comparação por ser frequentemente usado nos trabalhos relacionados.

No Gráfico 13, é possível observar que, da Etapa 1 à 5, os modelos baseados em árvore (*Random Forest* e LightGBM) apresentaram um RMSE menor em comparação aos modelos lineares, indicando melhor desempenho. Como explicado na seção anterior, essa

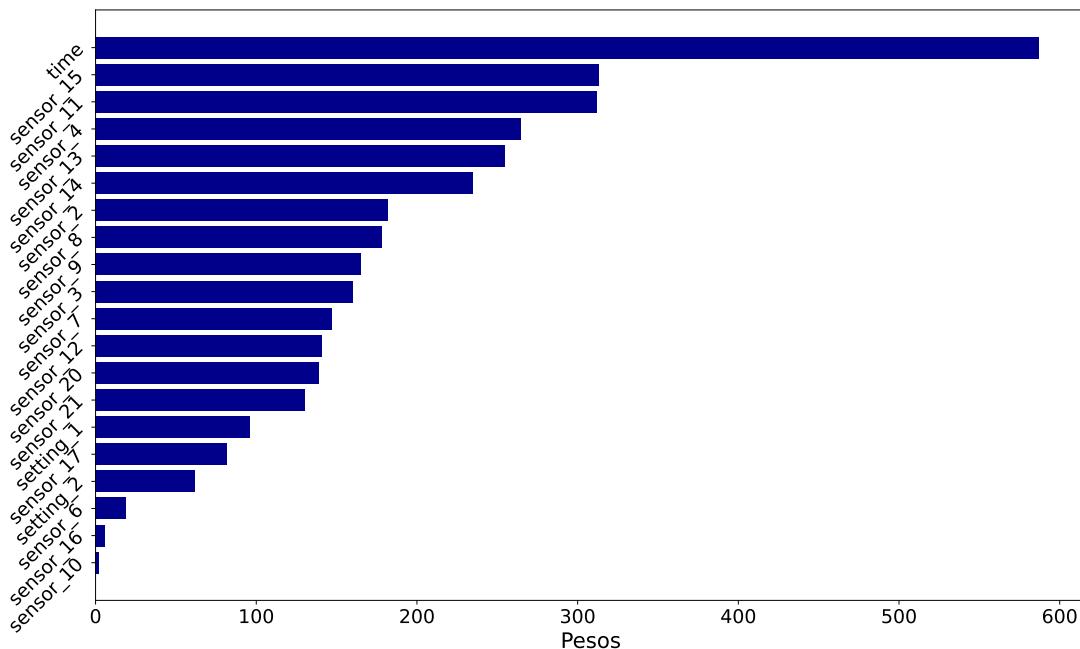
Gráfico 10 - Comparação entre os valores previstos pelo modelo LightGBM-etapa-7 e os valores reais da RUL no *dataset* FD002.



(a) Valores previstos *versus* real para o conjunto de treino.

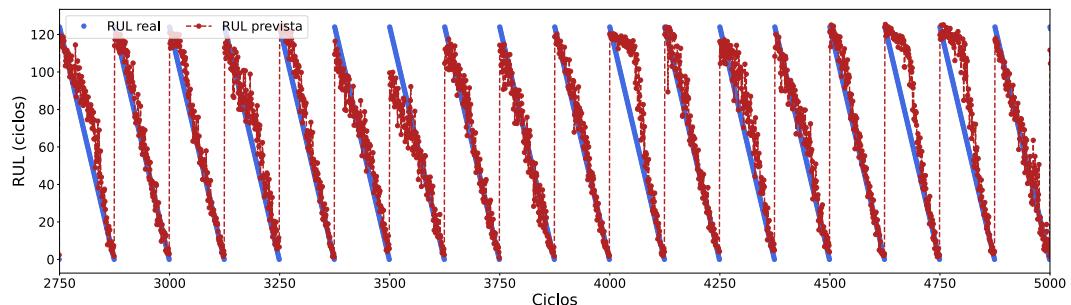


(b) Valores previstos *versus* real para o conjunto de teste.

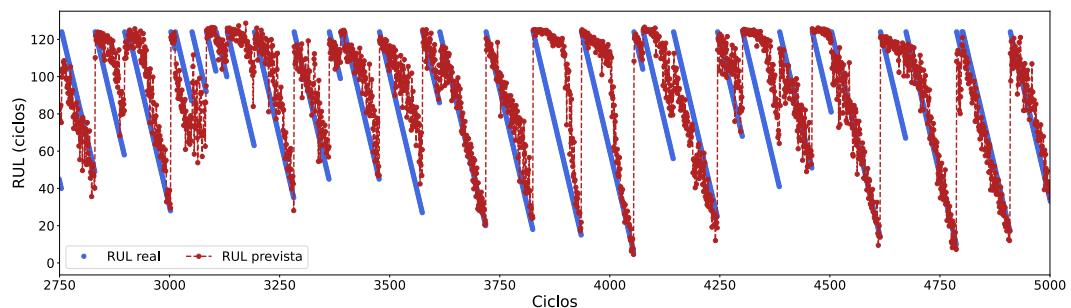


(c) Importância das variáveis.

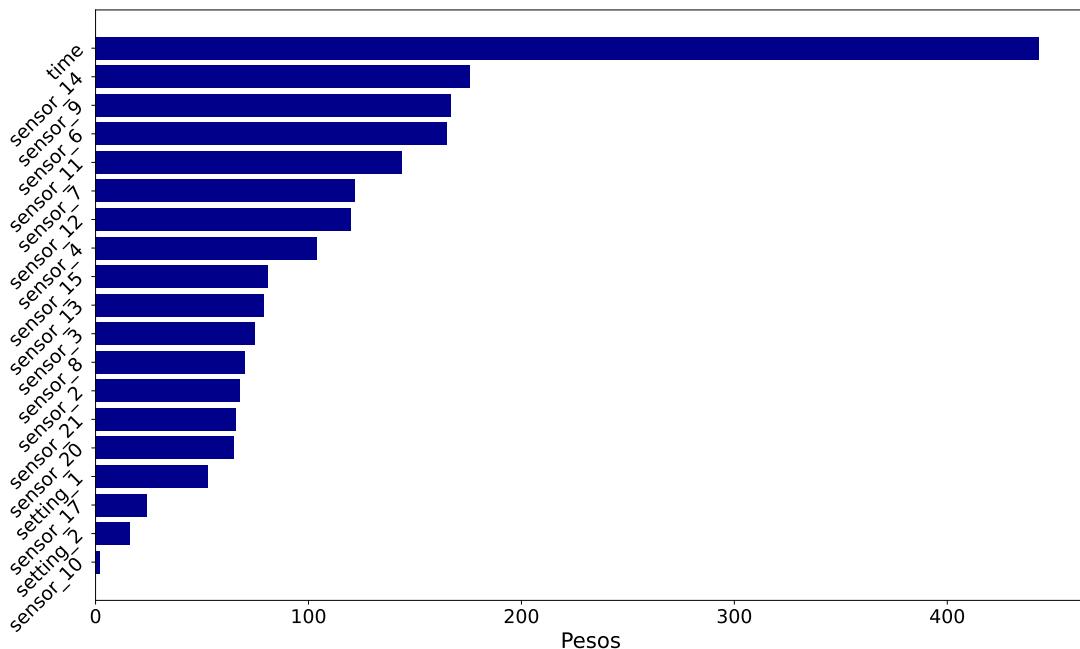
Gráfico 11 - Comparação entre os valores previstos pelo modelo LightGBM-etapa-7 e os valores reais da RUL no dataset FD003



(a) Valores previstos *versus* real para o conjunto de treino.

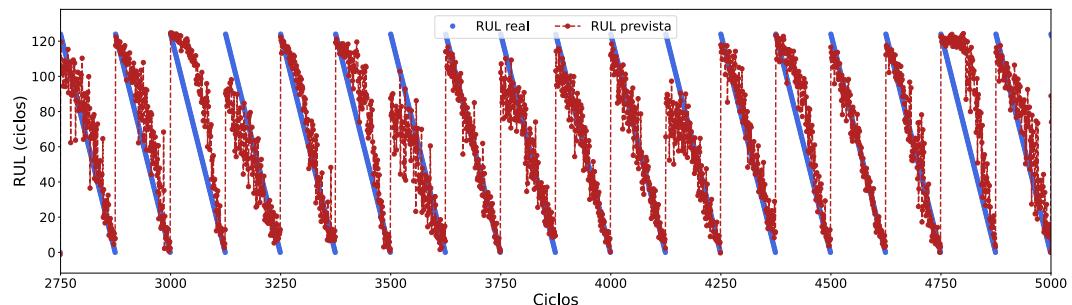


(b) Valores previstos *versus* real para o conjunto de teste.

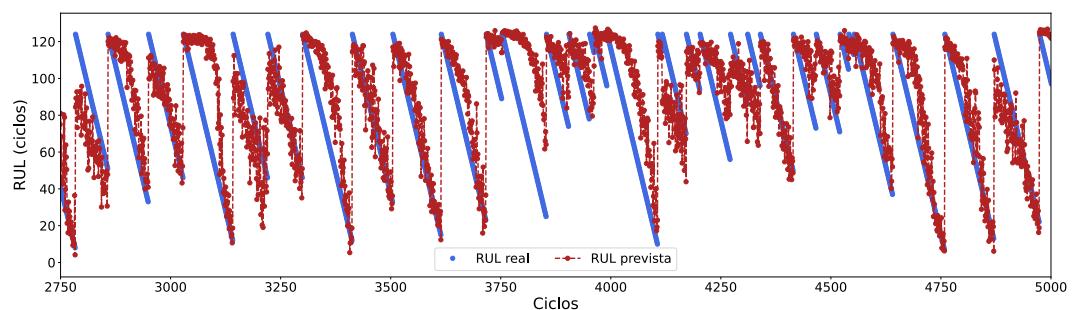


(c) Importância das variáveis.

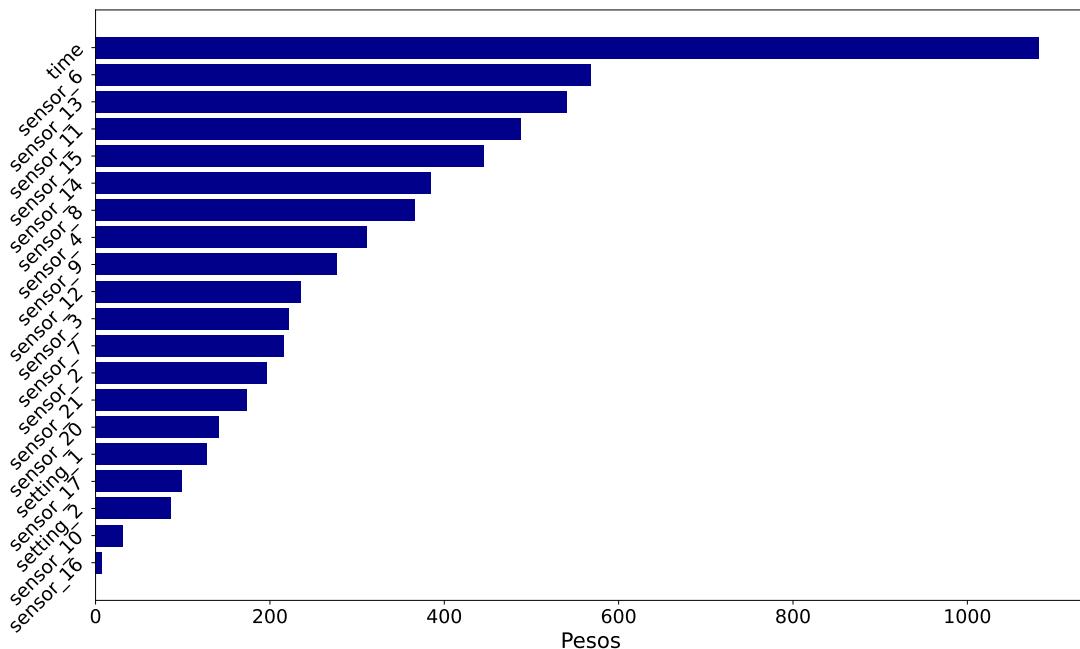
Gráfico 12 - Comparação entre os valores previstos pelo modelo LightGBM-etapa-7 e os valores reais da RUL no dataset FD004



(a) Valores previstos *versus* real para o conjunto de treino.

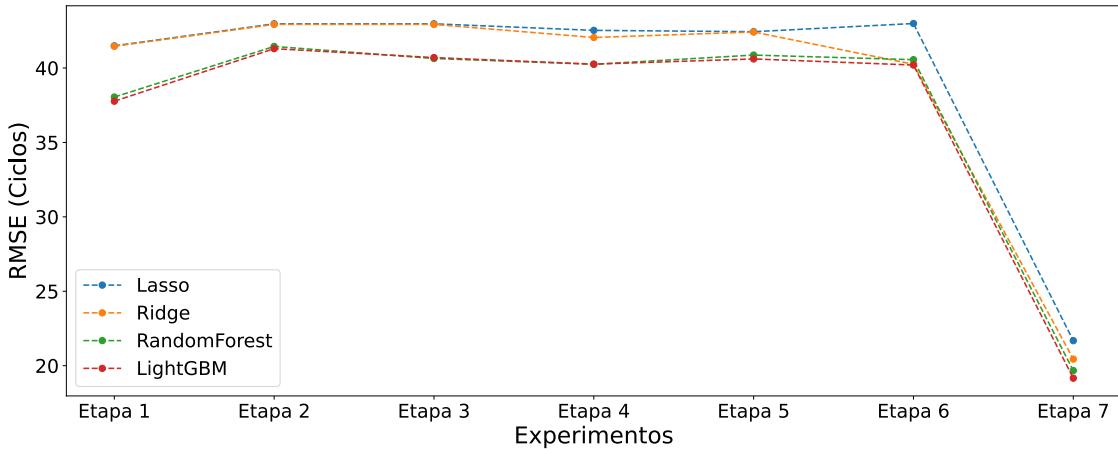


(b) Valores previstos *versus* real para o conjunto de teste.



(c) Importância das variáveis.

Gráfico 13 - Métricas para o conjunto de teste dos modelos em cada etapa no dataset FD001.



diferença era esperada devido à natureza não linear da degradação do motor Turbofan (PENG et al., 2021). No entanto, na Etapa 6, o modelo *Ridge* alcançou o mesmo desempenho que o *Random Forest* após a aplicação do pré-processamento PF, que introduziu dados não lineares nas entradas.

Na Etapa 7, houve uma melhoria significativa no RMSE, alcançando quase 50% de melhoria em todos os modelos. Contudo, o modelo LightGBM foi o que obteve a melhor métrica. É importante destacar que nesta etapa utilizou-se os dados na ROI, na qual a taxa de variação dos dados do motor Turbofan é mais elevada do que no início da campanha, proporcionando mais informações relevantes para o modelo.

Além do bom desempenho nas métricas, o modelo LightGBM também apresentou boa precisão nas previsões, como evidenciado no Gráfico 9b em comparação ao Gráfico 5b da Etapa 2. No Gráfico 9b, é possível notar que os valores previstos (pontos em vermelho) estão mais próximos dos valores reais (pontos em azul).

O modelo LightGBM também obteve as melhores métricas em todas as sete etapas, apesar das métricas do *Random Forest* terem sido próximas às do LightGBM. Embora sejam modelos baseados em árvore, o LightGBM possui características distintas, como algoritmo de *boosting*, GOSS e EFB, que o tornam superior. Em contraste, o *Random Forest* utiliza um algoritmo de *bagging*, o qual não conseguiu superar o desempenho do LightGBM. É importante destacar que o LightGBM foi treinado cinco vezes mais rápido que o *Random Forest* em alguns *datasets*. Portanto, o LightGBM demonstrou um desempenho global superior.

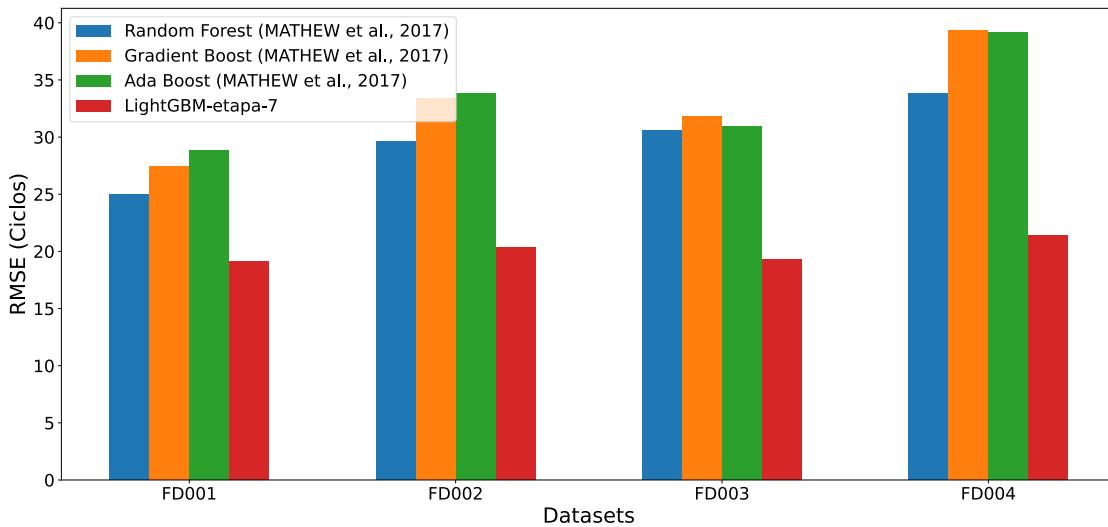
Vale destacar que o PyCaret já havia selecionado o LightGBM como o melhor modelo na Etapa 1, conforme a Tabela 3. No entanto, as métricas do modelo treinado pelo PyCaret foram inferiores ao modelo LightGBM treinado na Etapa 7, como pode ser

observado pela comparação das Etapas 1 e 7 no Gráfico 13.

Após realizar a análise comparativa dos modelos desenvolvidos neste trabalho, os próximos parágrafos irão se concentrar na comparação do melhor modelo obtido, o LightGBM-etapa-7, com os modelos desenvolvidos em trabalhos relacionados apresentados no Capítulo 3. Inicialmente, serão comparados com os modelos mais simples desenvolvidos por Mathew *et al.* (MATHEW et al., 2017). Em seguida, serão comparados com modelos mais complexos baseados em redes neurais desenvolvidos por Ellefsen *et al.* (ELLEFSEN et al., 2019), Li *et al.* (LI et al., 2018) e Babu *et al.* (BABU et al., 2016).

Mathew *et al.* (MATHEW et al., 2017) desenvolveram modelos de previsão de RUL para os quatro *datasets* utilizando as técnicas *Random Forest*, *Gradient Boost* e *Ada Boost*. Os resultados desses modelos foram comparados com o modelo proposto neste trabalho, conforme demonstrado no Gráfico 14. Observa-se que, para todos os *datasets*, o LightGBM-etapa-7 apresentou o menor RMSE, destacando-se como o melhor modelo entre os avaliados.

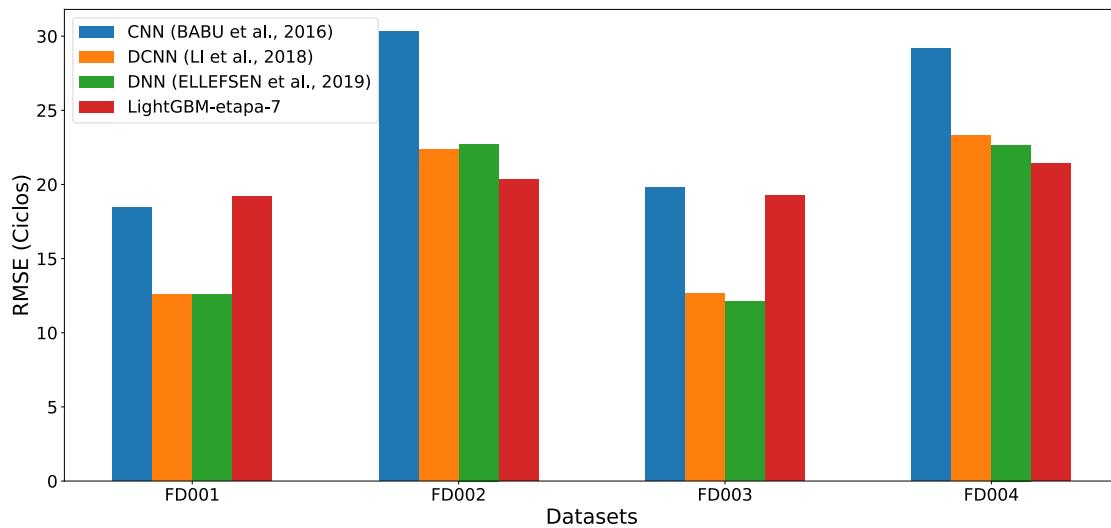
Gráfico 14 - Comparação do RMSE entre os modelos do Mathew *et al.* (MATHEW et al., 2017) e o modelo LightGBM-etapa-7.



O modelo LightGBM-etapa-7 foi comparado com modelos mais complexos baseados em redes neurais, tais como DNN (*Deep Neural Network*), DCNN (*Deep Convolutional Neural Network*) e CNN (*Convolutional Neural Network*), propostos por Ellefsen *et al.* (ELLEFSEN et al., 2019), Li *et al.* (LI et al., 2018) e Babu *et al.* (BABU et al., 2016), respectivamente. As métricas desses modelos foram comparadas com a métrica do LightGBM-etapa-7 e apresentadas no Gráfico 15. Esse gráfico mostra que, nos *datasets* pares (FD002 e FD004), o modelo LightGBM-etapa-7 superou os modelos baseados em

redes neurais, ou seja, ele apresentou o menor RMSE. É importante destacar que o modelo modelo CNN obteve um RMSE de 30,29 para o *dataset* FD002, enquanto o modelo proposto obteve um RMSE de 20,34, representando uma melhoria de 32% pelo LightGBM-etapa-7. Em contrapartida, nos *datasets* ímpares (FD001 e FD003), os modelos DNN e DCNN apresentaram o melhor desempenho. Mesmo assim, o LightGBM-etapa-7 se igualou aos resultados do modelo CNN. Conforme mencionado na Seção 4.1, os datasets ímpares compartilham o mesmo nível de desgaste inicial, e o mesmo acontece com os conjuntos de dados pares. Os resultados do Gráfico 15 demonstram essa característica dos dados já que as métricas desses conjuntos foram semelhantes.

Gráfico 15 - Comparaçāo do RMSE entre os modelos em redes neurais dos trabalhos relacionados e o modelo LightGBM-etapa-7.



CONCLUSÕES

Este trabalho mostrou a importância da PdM tanto para a segurança quanto para o sucesso financeiro das empresas. Além disso, foram apresentadas ferramentas de código aberto que podem ser utilizadas para aprimorar a tomada de decisão em relação à PdM, tais como modelos de aprendizado de máquina. Foi possível observar que cada modelo possui características únicas e que é importante selecionar o modelo mais adequado para um conjunto específico de dados. Por exemplo, modelos lineares não são indicados para prever a RUL do Turbofan, uma vez que sua degradação não é linear.

O modelo de aprendizado de máquina utilizado neste trabalho foi desenvolvido por meio da técnica de modelagem incremental, na qual cada experimento agregou na melhoria do modelo em cada etapa. Como resultado, foi obtido o melhor modelo, o LightGBM-etapa-7, que superou modelos mais complexos, como os modelos em redes neurais, na previsão dos datasets FD002 e FD004. Esse resultado evidencia que um pré-processamento eficaz é crucial para melhorar significativamente os resultados do modelo. No entanto, nem todas as técnicas de pré-processamento utilizadas foram benéficas para o modelo. Por exemplo, ao aplicar a suavização nos dados do Turbofan com SG, as métricas dos modelos foram reduzidas. Isso ocorreu porque o ruído presente nos dados continha informações relevantes para o modelo.

Os experimentos da Etapa 7 garantiram que os modelos tivessem uma boa assertividade na previsão da RUL no final da campanha. É fundamental que os engenheiros de manutenção tenham um modelo mais assertivo para o final da vida útil do equipamento, uma vez que essa fase é crítica e requer um planejamento cuidadoso da manutenção. Durante esse período, o equipamento precisa passar por inspeções e manutenções rigorosas. Portanto, um modelo com maior precisão nessa fase pode ajudar a garantir a segurança da operação e a reduzir os custos associados à manutenção e reparos.

A utilização do modelo desenvolvido neste trabalho tem o potencial de aprimorar a manutenção preditiva (PdM) de motores Turbofan e outros equipamentos, proporcionando às empresas a capacidade de planejar antecipadamente a manutenção, reduzir custos e prevenir acidentes. Com o intuito de aprimorar ainda mais as métricas de desempenho, existem algumas direções para trabalhos futuros.

Uma próxima etapa consiste em desenvolver um modelo de redes neurais utilizando o mesmo pré-processamento de dados adotado neste trabalho. Além disso, será aplicada uma técnica de pré-processamento que envolve a correção dinâmica do *bias*, com o objetivo de reduzir a variância e o desvio do modelo. Adicionalmente, outros algoritmos, como *Elastic Net* e Redes Siamesas, serão testados e comparados com os resultados obtidos.

Também será explorado o uso de métodos de controle estatístico, como as cartas de controle, para monitorar anomalias no sistema. Essa abordagem permitirá identificar des-

vios significativos em relação ao comportamento esperado, contribuindo para a detecção precoce de falhas e a tomada de medidas corretivas.

Além disso, serão desenvolvidos dois modelos baseados nos experimentos realizados na etapa 7. O primeiro modelo será responsável por realizar previsões de RUL acima de 125 ciclos, enquanto o segundo modelo se concentrará na previsão da RUL abaixo desse limite. Essa abordagem permitirá uma análise mais detalhada e uma compreensão aprimorada das condições de funcionamento do motor ao longo de sua vida útil.

REFERÊNCIAS

- AVEVA. *Predictive Analytics*. 2023. Disponível em: <<https://www.aveva.com/en/products/predictive-analytics/>>. Acesso em: 14 de mar. 2023.
- BABU, Giduthuri Sateesh et al. Deep convolutional neural network based regression approach for estimation of remaining useful life. Springer International Publishing, Cham, p. 214–228, 2016. Disponível em: <<https://personal.ntu.edu.sg/xlli/publication/RUL.pdf>>. Acesso em: 28 de mar. 2023.
- BARBOSA, Tiudorico Leite. Um histórico da manutenção e conceitos sobre sua função. Revista Marítima Brasileira, 2018. Disponível em: <<http://portaldeperiodicos.marinha.mil.br/index.php/revistamaritima/article/view/173/150>>. Acesso em: 23 fev. 2023.
- BLANCHARD, Benjamin S. et al. *Maintainability: A Key to Effective Serviceability and Maintenance Management*. 1. ed. [S.l.]: Wiley-Interscience, 1995. 560 p.
- BROWN, Sara. *Machine learning, explained*. 2021. Disponível em: <<https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained>>. Acesso em: 16 de mar. 2023.
- CHICCO, Davide et al. The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation. *PeerJ*, 2021. Disponível em: <<https://peerj.com/articles/cs-623/>>. Acesso em: 23 de mar. 2023.
- ELLEFSEN, André Listou et al. Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture. *Reliability Engineering e System Safety*, v. 183, p. 240–251, 2019. ISSN 0951-8320. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0951832018307506>>. Acesso em: 28 de mar. 2023.
- FACELI, Katti et al. *Inteligência Artificial: Uma abordagem de aprendizado de máquina*. 1. ed. Rio de Janeiro: LTC, 2011. 396 p. ISBN 978-85-216-1880-5.
- GE do Brasil. *A arte da engenharia: o maior motor a jato do mundo*. 2016. Disponível em: <<https://greportsbrasil.com.br/a-arte-da-engenharia-o-maior-motor-a-jato>>. Acesso em: 14 de mar. 2023.
- General Eletrics. *SmartSignal analytics*. 2023. Disponível em: <<https://www.ge.com/digital/applications/asset-performance-management-apm-software/equipment-downtime-prevention-ge-smartsignal>>. Acesso em: 14 de mar. 2023.
- GÉRON, Aurélien. *Mãos à obra: aprendizado de máquina com Scikit-Learn e TensorFlow*. 1. ed. Rio de Janeiro: Starlin Alta Editora e Consultoria Eireli, 2019. 579 p.
- GOSWAMI, Manash. *What is automated machine learning (AutoML)?* 2023. Disponível em: <<https://learn.microsoft.com/en-us/azure/machine-learning/concept-automated-ml>>. Acesso em: 22 de mar. 2023.
- GRANATYR, Jones; PACHOLOK, Edson. *The Ultimate Beginner's Guide to Genetic Algorithms in Python*. 2022. Curso online. Disponível em: <<https://www.udemy.com/course/the-ultimate-beginners-guide-to-genetic-algorithms-in-python/>>.

- HALL, Nancy. *Turbofan Engine*. Glenn Research Center, 2021. Disponível em: <<https://www.grc.nasa.gov/www/k-12/airplane/aturbf.html>>. Acesso em: 29 dez. 2022.
- HASTIE, Trevor et al. *The elements of statistical learning: data mining, inference and prediction*. 2. ed. Springer, 2009. Disponível em: <<https://hastie.su.domains/Papers/ESLII.pdf>>. Acesso em: 20 de mar. 2023.
- HEIMES, Felix O. Recurrent neural networks for remaining useful life estimation. p. 1–6, 2008.
- HOERL, Arthur E. et al. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, Taylor e Francis, Ltd., American Statistical Association, American Society for Quality, v. 12, n. 1, p. 55–67, 1970. ISSN 00401706. Disponível em: <<http://www.jstor.org/stable/1267351>>. Acesso em: 20 de mar. 2023.
- HOMEM, William Ludovico et al. Estimativa do tempo de vida útil para motores aeronáuticos comerciais: uma abordagem recorrente e convolucional. *Ifes Cência*, 2020. Disponível em: <<https://ojs.ifes.edu.br/index.php/dect/article/view/761>>. Acesso em: 28 de mar. 2023.
- IBM. *IBM Global AI Adoption Index 2022*. 2022. Disponível em: <<https://www.ibm.com/downloads/cas/GVAGA3JP>>. Acesso em: 16 de mar. 2023.
- ISO 55000. *Asset management: Overview, principles and terminology*. Geneva, CH, 2014. Disponível em: <<https://www.iso.org/obp/ui/#iso:std:iso:55000:ed-1:v2:en>>.
- JÚNIOR, Wagner José de Alvarenga. *Métodos de otimização hiperparamétrica: um estudo comparativo utilizando árvores de decisão e florestas aleatórias na classificação binária*. 2018. Dissertação (Mestrado) — Universidade Federal de Minas Gerais, Minas Gerais, 2018. Disponível em: <<http://hdl.handle.net/1843/BUBD-AX2NLF>>. Acesso em: 23 de mar. 2023.
- KARDEC, Alan et al. *Manutenção: Função Estratégica*. 1. ed. [S.l.]: Quality Mark, 2001. 341 p.
- KE, Guolin et al. Lightgbm: A highly efficient gradient boosting decision tree. In: . Red Hook, NY, USA: Curran Associates Inc., 2017. (NIPS'17), p. 31493157. ISBN 9781510860964. Disponível em: <<https://dl.acm.org/doi/10.5555/3294996.3295074>>. Acesso em: 22 de mar. 2023.
- KHILLAR, Sagar. *Difference Between Bagging and Random Forest*. 2019. Disponível em: <<http://www.differencebetween.net/technology/difference-between-bagging-and-random-forest/>>. Acesso em: 21 de mar. 2023.
- LI, Xiang et al. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering e System Safety*, v. 172, p. 1–11, 2018. ISSN 0951-8320. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0951832017307779>>.
- LIDIANE, Lordelo Mendes Kruschewsky et al. Análise fatorial por meio da matriz de correlação de pearson e policórica no campo das cisternas. *S Engineering and Science*,

v. 7, n. 1, p. 58–70, abr. 2018. Disponível em: <<https://periodicoscientificos.ufmt.br/ojs/index.php/eng/article/view/5266>>.

MARTINS, Ricardo Bohadana. *Desenvolvimento de solução computacional baseada em machine learning para apoio à manutenção preditiva*. Niterói: Universidade Federal Fluminense (UFF), 2022. 82 p. Disponível em: <https://github.com/ricardobohadana/PredictiveMaintenance/blob/main/RICARDO_TCC_UFF_V5_FINAL_COMPLETO.pdf>. Acesso em: 27 dez. 2022.

MATHEW, Vimala et al. Prediction of remaining useful lifetime (rul) of turbofan engine using machine learning. IEEE, International Conference on Circuits and Systems (ICCS), p. 306–311, 2017. Disponível em: <<https://ieeexplore.ieee.org/document/8326010/authors#authors>>. Acesso em: 28 dez. 2022.

MUSSO, Marianna. *Diferença entre gestão de ativos e da manutenção*. 2022. Disponível em: <<https://tractian.com/blog/diferenca-gestao-ativos-manutencao>>. Acesso em: 01 de mar. 2023.

NBR 5462. *Confiabilidade e mantinabilidade*. Rio de Janeiro - RJ, 1994. Disponível em: <<https://www.normas.com.br/visualizar/abnt-nbr-nm/8044/nbr5462-confiabilidade-e-mantenabilidade>>.

NISHIDA, Erica. *Propriedades da Filtragem de Savitzky-Golay Aplicadas na Identificação de Complexos QRS em Sinais de Eletrocardiograma*. 2017. 53 f. Dissertação (Mestrado) — Universidade Federal de Itajubá, Minas Gerais, 2017. Disponível em: <https://repositorio.unifei.edu.br/xmlui/bitstream/handle/123456789/954/dissertacao_nishida_2017.pdf>. Acesso em: 16 de mar. 2023.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

PENG, Cheng et al. A remaining useful life prognosis of turbofan engine using temporal and spatial feature fusion. MDPI, v. 21, n. 418, p. 1–9, 2021. Disponível em: <<https://doi.org/10.3390/s21020418>>. Acesso em: 28 dez. 2022.

PINTO, Matheus Henrique Tavares. *Seleção de variáveis usando o algoritmo genético*. 2022. 55 f. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, Rio Grande do Norte, 2022. Disponível em: <https://repositorio.ufrn.br/bitstream/123456789/48411/1/Selecaovariaveisusando_Pinto_2022.pdf>. Acesso em: 19 de mar. 2023.

SANTOS, Gustavo. *Introdução à árvore de decisão*. 2022. Disponível em: <<https://medium.com/data-hackers/introdu%C3%A7%C3%A3o-%C3%A0-%C3%A1rvore-de-regress%C3%A3o-3845c8eba857>>. Acesso em: 20 de mar. 2023.

SAXENA, Abhinav et al. Damage propagation modeling for aircraft engine run-to-failure simulation. IEEE, International Conference on Prognostics and Health Management, p. 1–9, 2008. Disponível em: <<https://ieeexplore.ieee.org/document/4711414>>. Acesso em: 29 dez. 2022.

SILVA, Aliecha et al. *Como Funciona uma Turbina de Avião? Motor a Reação Chamado de Turbofan*. 2017. Disponível em: <<https://www.youtube.com/watch?v=eglDumaJeEg>>. Acesso em: 15 de mar. 2023.

TEUBERT, Christopher. *CMAPSS Jet Engine Simulated Data*. Prognostics Center of Excellence Data Set Repository of NASA, 2007. Disponível em: <<https://data.nasa.gov/Aerospace/CMAPSS-Jet-Engine-Simulated-Data/ff5v-kuh6>>. Acesso em: 28 dez. 2022.

THAKKAR, Unnati et al. Remaining useful life prediction of an aircraft turbofan engine using deep layer recurrent neural networks. *Actuators*, v. 11, n. 3, 2022. ISSN 2076-0825. Disponível em: <<https://www.mdpi.com/2076-0825/11/3/67>>.

Tractian. *O salto na manutenção de ativos na Embraer*. 2021. Disponível em: <<https://tractian.com/blog/o-salto-da-confiabilidade-e-disponibilidade-de-ativos-na-embraer-com-a-solucao-da-tractian>>. Acesso em: 14 de mar. 2023.

VASCONCELOS, Bruno Freitas Boynad de. *Poder preditivo de métodos de Machine Learning com processos de seleção de variáveis: uma aplicação às projeções de produto de países*. 2017. 55 f. Dissertação (Doutorado) — Universidade de Brasília, Brasília, 2017. Disponível em: <https://repositorio.unb.br/bitstream/10482/23995/1/2017_BrunoFreitasBoynarddeVasconcelos.pdf>. Acesso em: 20 de mar. 2023.

YUREK, Ozlem Ece; BIRANT, Derya. *Remaining Useful Life Estimation for Predictive Maintenance Using Feature Engineering*. IEEE, Innovations in Intelligent Systems and Applications Conference (ASYU), 2019. 1-5 p. Disponível em: <<https://ieeexplore.ieee.org/document/8946397>>. Acesso em: 27 dez. 2022.

ZAMPOLLI, Marissa et al. *Gestão de ativos: guia para a aplicação da norma ABNT NBR ISO 55001 considerando as diretrizes da ISO 55002:2018*. International Copper Association Brazil, 2019. Disponível em: <<https://www.leonardo-energy.org.br/wp-content/uploads/2019/10/gestao-de-ativos-guia-para-a-aplicacao-da-iso-55001.pdf>>. Acesso em: 01 de mar. 2023.