Aplicação de Modelos de Redes Neurais na Manutenção Preditiva em Sistemas Industriais

Arthur Souto Lima[1]

[1]Departamento de Ciência da Computação (DCC)
Universidade Federal de Minas Gerais (UFMG)
Av. Pres. Antônio Carlos, 6627 - Pampulha, Belo Horizonte - MG, 31270-901

arthursl@ufmg.br

Abstract. *Following the transformations brought by Industry 4.0, predictive maintenance is a field of study aiming to improve industrial maintenance. One parameter frequently studied is the Remaining Useful Life (RUL) of a system or a machine. With this purpose, neural networks using different strategies were proposed and trained over NASA's turbofan engine dataset in order to predict RUL from sensor data. Three-layer dense architectures stood out due to their good performance across various scenarios. Moreover, in order to improve model explainability, analyses were made using the SHAP library.*

Resumo. *Na esteira das transformações da Indústria 4.0, a manutenção preditiva é um dos ramos de estudo para melhoria dos processos de manutenção industrial e um dos parâmetros recorrentemente estudados é a Vida Útil Restante (RUL, Remaining Useful Life) de um sistema ou equipamento. Com esse objetivo, foram propostas arquiteturas de redes neurais com diferentes estratégiase treinadas no dataset de motores turbofan da NASA, a fim de se estimar a RUL com base nos dados de vários sensores. As redes densas com três camadas tiveram um desempenho destacado destaque devido a uma boa performance nos vários cenários. Por fim, visando aumentar a explicabilidade dos modelos treinados, foram feitas análises a partir da biblioteca SHAP.*

1. Introdução

Os recentes avanços industriais trazidos pelo contexto da Indústria 4.0 abarcam uma série de mudanças advindas da implantação de novas tecnologias. Essa nova fase trouxe benefícios, mas também novos desafios a serem investigados e superados. Com integração de outras áreas do conhecimento, como a Ciência da Computação, principalmente às Engenharias de Produção e Automação, são possibilitadas novas abordagens no contexto industrial, por exemplo no âmbito da comunicação, com a Internet das Coisas (IoT, Internet of Things), ou no setor de projetos, com sistemas cyberfísicos [Peres et al. 2020].

Nesse contexto, a inteligência artificial (IA) também se mostra como um catalisador de soluções. As técnicas e métodos de IA garantem aos sistemas e aos processos uma capacidade de interagir com o ambiente em que se encontram, adquirindo dados e resolvendo problemas complexos, juntamente com a possibilidade de aprenderem com a experiência e se tornarem melhores em suas respectivas tarefas [Peres et al. 2020]. Nesse momento de recente introdução dessas abordagens ao setor industrial, no âmbito da Indústria 4.0, devido à grande produção de dados nos mais diversos sistemas integrados,

são necessárias soluções focadas neles [Angelopoulos et al. 2020], as quais usualmente advém do aprendizado de máquina.

O aprendizado de máquina (ML, machine learning) é uma sub-área da inteligência artificial, centrada nos dados, os quais servem de base para criação de modelos, a fim de se realizar análises e previsões [Russell and Norvig 2020]. Todos os dias surgem novas estratégias e formas de se refinar esses modelos, bem como novos campos de aplicação, inclusive o industrial, mais recentemente [Peres et al. 2020]. As técnicas e métodos desse campo, juntamente com a maior capacidade de processamento possibilitado pela tecnologia atual, permitem abordagens via aprendizado supervisionado, semi-supervisionado, não-supervisionado, aprendizado por reforço e, ainda, aprendizado profundo [Jagatheesaperumal et al. 2021].

No aprendizado supervisionado, há diversos sistemas que, por exemplo, usam algoritmos como tradicionais Support Vector Machine (SVM) e Random Forest Classifier, com resultados expressivos [Peres et al. 2020, Silvestrin et al. 2019]. O aprendizado profundo também propõe suas soluções, sejam elas baseadas em métodos de redes neurais artificiais (ANN), convolucionais (CNN) ou profundas, já no contexto do deep learning (DL) [Silvestrin et al. 2019, Serradilla et al. 2022].

Outra abordagem recente advinda da aplicação de outros conhecimentos na indústria está relacionada à manutenção. Na literatura, pode-se elencar três grandes tipos de manutenção [Carvalho et al. 2019]: a manutenção corretiva ou Run-to-Failure (R2F), a manutenção preventiva (PvM,preventive maintenance) e a manutenção preditiva (PdM, predictive maintenance). A primeira é a manutenção mais simples, pois as correções só são aplicadas quando há alguma falha. No outro lado do espectro, a manutenção preventiva utiliza um raciocínio que programa as revisões periodicamente para evitar as quebras ou as falhas.

O terceiro tipo se vale de estratégias baseadas na análise de dados. Usando ferramentas, estima-se quando será necessária uma manutenção ou mesmo quanto de vida útil aquele equipamento ainda possui antes de ser preciso uma substituição ou manutenção. Desse modo, técnicas estatísticas e de aprendizado de máquina permitem uma abordagem para alcançar essas soluções, no intuito de maximizar o tempo de operação das linhas produtivas [Silvestrin et al. 2019, Zonta et al. 2020].

A adoção, paulatina e gradual, dessas técnicas nesse contexto industrial traz uma série de benefícios a diversos agentes envolvidos direta ou indiretamente, nos vários setores em que essa integração pode ser observada [Peres et al. 2020, Jagatheesaperumal et al. 2021]. No âmbito de uma produção preditiva, pode-se prever as necessidades dos clientes e do mercado a priori e adaptar a produção a isso, retornando mais lucro à empresa. Por outro lado, com a manutenção preditiva, há uma melhora na confiabilidade dos equipamentos e dos sistemas, o que otimiza os custos com reparos e garante mais segurança a todos os envolvidos com a produção[Jagatheesaperumal et al. 2021].

1.1. Problema e Objetivos

No âmbito da manutenção preditiva, tem-se a área de Prognósticos e Gerência de Saúde (PHM, Prognostics and Health Management), a qual se dedica a estimar o estado do equipamento ou do sistema num momento posterior [Leser 2017, Wang et al. 2020]. Mais

precisamente, é necessário saber em que momento o sistema não estará mais em um estado de funcionamento, ou quanto tempo até que isso ocorra, e essa informação é essencial para o planejamento das manutenções [Berghout and Benbouzid 2022]. A estimativa, portanto, da Remaining Useful Life (RUL, vida útil restante) pode ser feita a partir de modelos físicos, a partir de dados ou com base numa estratégia híbrida [Wang et al. 2020].

Na abordagem a partir dos dados, as redes neurais e o aprendizado profundo apresentam diversos algoritmos e arquiteturas, com diferentes vantagens e desvantagens. Por isso, é importante investigar e compreender alguns modelos de redes neurais para a estimativa da RUL.

Além disso, como ressaltam [Peres et al. 2020, Ferreira and Gonçalves 2022], é relevante que os modelos usados para previsões na indústria não sejam aplicados apenas como uma ”caixa-preta”, isto é, seus resultados devem ter uma interpretabilidade e explicabilidade. Essas exigências podem advir de medidas regulatórias ou mesmo pela necessidade de uma maior transparência desses métodos e, portanto, devem ser uma questão de importância no projeto também.

2. Trabalhos Relacionados

A manutenção preditiva é um assunto bastante estudado atualmente, com novas abordagens e métodos sendo propostos recorrentemente. Artigos recentes usaram aprendizado de máquina para avaliar a estabilidade de sistemas hidráulicos a partir de árvores de decisão e redes neurais [Silvestrin et al. 2019], manutenção em veículos usando Random Forests [Prytz et al. 2015] ou mesmo estratégias usando uma miríade de algoritmos [Susto et al. 2015].

Na área de PHM, um parâmetro muito comum de ser estimado é a RUL e, em abordagens dirigidas por dados, arquiteturas de redes neurais recorrentes são bastante utilizadas, como por exemplo os baseados em Long Short Term Memory (LSTM), tal qual visto em [Silvestrin et al. 2019]. Além disso, nota-se o uso de redes neurais convencionais e convolucionais para tal tarefa em [Wang et al. 2020], a partir de diversos cenários.

Dentre os conjuntos de dados usados normalmente para estudos da área, o dataset de motores turbofan da NASA [Saxena et al. 2008] é bastante utilizado em vários deles. Com ele, usualmente a tarefa é de estimar a RUL a partir dos dados dos sensores, seja, por exemplo, por métodos baseados em aprendizado profundo ou em redes de várias camadas de percéptrons [Li et al. 2018, Yuan et al. 2016]. Pode-se também fazer uma abordagem com métodos tradicionais como Random Forest e compará-lo com estratégias de DL [Mathew et al. 2017, Berghout and Benbouzid 2022].

3. Metodologia

3.1. Visão Geral

O projeto segue o fluxo de trabalho de aprendizado de máquina. Inicialmente, separa-se os dados em treino e teste e usa-se apenas o primeiro grupo para elaborar e refinar os modelos. As primeiras investigações são centradas numa análise mais exploratória das informações, para, em seguida, focar-se na implementação dos modelos em si. Em todos eles, buscou-se realizar uma sintonia de seus hiperparâmetros visando melhores resultados e, além disso, fez-se a validação cruzada a fim de aperfeiçoar seu poder de

generalização. Com os eles já finalizados, conduz-se uma etapa de análise dos mesmos, visando incrementar sua interpretabilidade e explicabilidade.

Considerando o desenvolvimento da primeira etapa do projeto, bastante do ferramental utilizado foi reaproveitado, sobretudo no que tange à interface com os dados e seu processamento inicial. Além disso, o arcabouço de classes e métodos para análise dos resultados e da explicabilidade apenas precisaram de menores adaptações para a nova situação dos modelos baseados em redes neurais.

A sintonia dos hiperparâmetros nessa etapa se deu via Bayes Search, uma estratégia que faz uma otimização bayesiana no espaço de hiperparâmetros a fim de se realizar uma busca mais eficiente dos melhores [Wu et al. 2019]. Dessa forma, iterativamente, os modelos eram refinados seguindo a otimização no intuito seja de minimizar as métricas de erro ou de maximizar as de acurácia. A escolha desse método de sintonia se deu devido, principalmente, à grande quantidade de hiperparâmetros, sendo muitos deles com grandes intervalos de busca, como a quantidade de neurônios de uma camada ou as proporções de dropout e de validação interna. Essa estratégia permitiu que mesmo modelos mais complexos possam ser otimizados em um horizonte de tempo de horas, treinando em todo o conjunto de dados a cada rodada.

Tendo em vista que as tarefas de aprendizado são de regressão, escolheu-se como métricas de avaliação dos modelos o coeficiente de determinação, ou R^2, e também a raiz do erro quadrado médio (RMSE, Root Mean Squared Error). A primeira é calculada diretamente pelas ferramentas usadas. A segunda normalmente é disponibilizada como erro quadrado médio e, por isso, exigiu uma adaptação para que fosse usada a raiz dele.

As arquiteturas básicas escolhidas para a composição nos modelos a serem investigados foram, no amplo contexto de redes neurais artificiais (ANNs, Artificial Neural Networks), os Multi-Layer Perceptrons (MLPs) e as LSTMs, já no âmbito das redes recorrentes (RNNs, Recurrent Neural Networks). Além deles, advêm, da primeira etapa do projeto, os resultados do Dummy Regressor que serve de desempenho-base de referência e também a da RF que obteve melhor performance naquele momento. Tal qual anteriormente, também se analisará a influência da adição de features polinomiais (Polynomial Features) dos atributos de entrada para as ANNs.

Em termos de ferramentas, o código foi escrito em Python e executado em forma de notebooks Jupyter no Google Colab. Para a estruturação dos dados, usou-se a biblioteca Pandas [Wes McKinney 2010] e a NumPy [Harris et al. 2020]. Para os modelos, foi utilizada a biblioteca Scikit-Learn [Pedregosa et al. 2011], juntamente com Tensorflow[TensorFlow Developers 2022] e Keras [Chollet et al. 2015]. Para a otimização bayesiana, tem-se a biblioteca scikit-optimize [Head et al. 2020]. Para as análises de explicabilidade, a biblioteca SHAP (SHapley Additive exPlanations) [Lundberg and Lee 2017]. Por fim, em relação aos gráficos e artefatos visuais, a biblioteca Matplotlib [Hunter 2007].

3.2. Arquiteturas Propostas

Diferentemente do aprendizado de máquina tradicional, em que os modelos já estão implementados, bastando instanciá-los e sintonizar seus hiperparâmetros, no âmbito das redes neurais há a necessidade de se elaborar a arquitetura do modelo, tarefa de suma relevância para seu bom desempenho. No projeto serão usados dois tipos de camadas nas redes: camadas densas ou totalmente conectadas e camadas LSTM.

No projeto foram implementadas também classes de modelos customizados do scikit-learn como invólucros dos modelos em si, promovendo um encapsulamento e servindo de interface para os métodos e funções tanto do Tensorflow-Keras quanto do scikit-learn. Isso facilitou o pré-processamento dos dados, bem como a análise dos resultados, já que os métodos para obter as métricas de desempenho também foram implementadas nessas classes. Desse modo, de certa forma, no uso das redes propostas teve-se a sensação de uma ”caixa-preta”semelhante ao aprendizado tradicional, contudo, no caso, os componentes foram implementados no decorrer do projeto e conhecia-se seu funcionamento.

A fim de ilustrar as arquiteturas propostas, foram elaborados diagramas: na figura 1 com a representação das ANNs e na 2 as aquelas baseadas em LSTMs. Convém ressaltar que elas são apenas ilustrativas, ou seja, no caso das ANNs, é possível que as camadas internas tenham diferentes tamanhos, mas foram representadas de forma igual para fins de apresentação apenas.

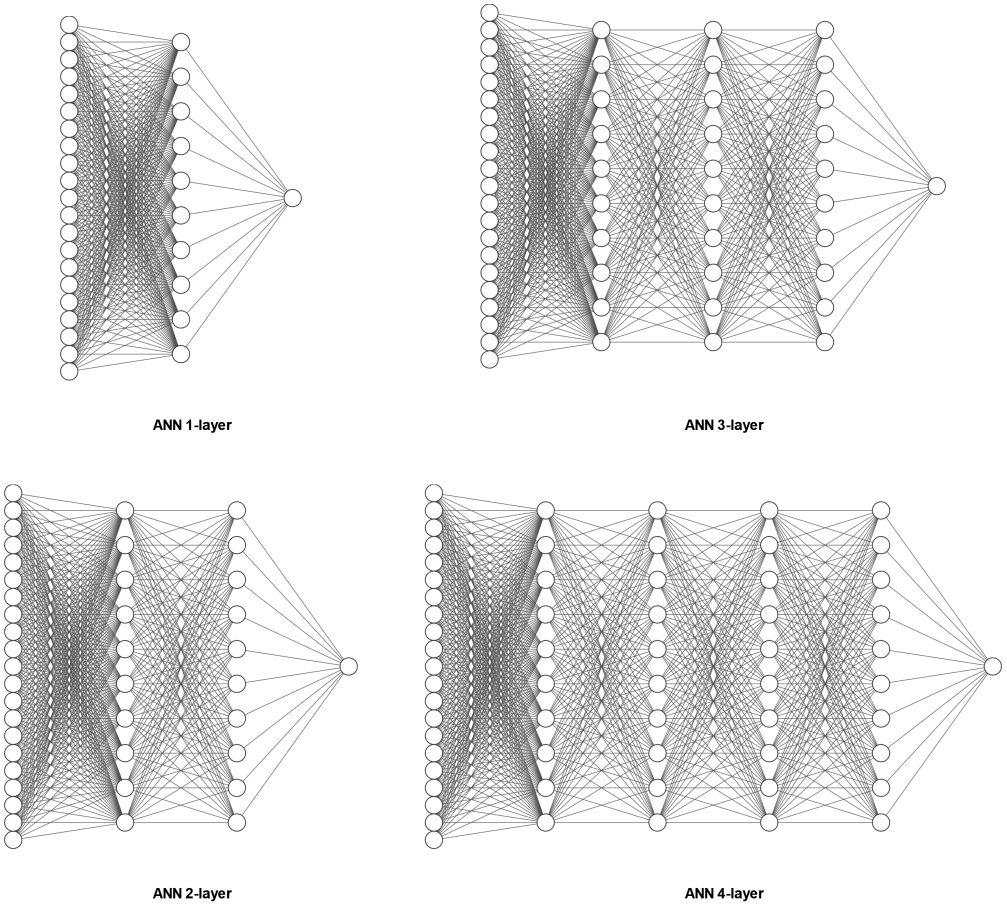

Figura 1. Esquema representativo das arquiteturas das ANNs

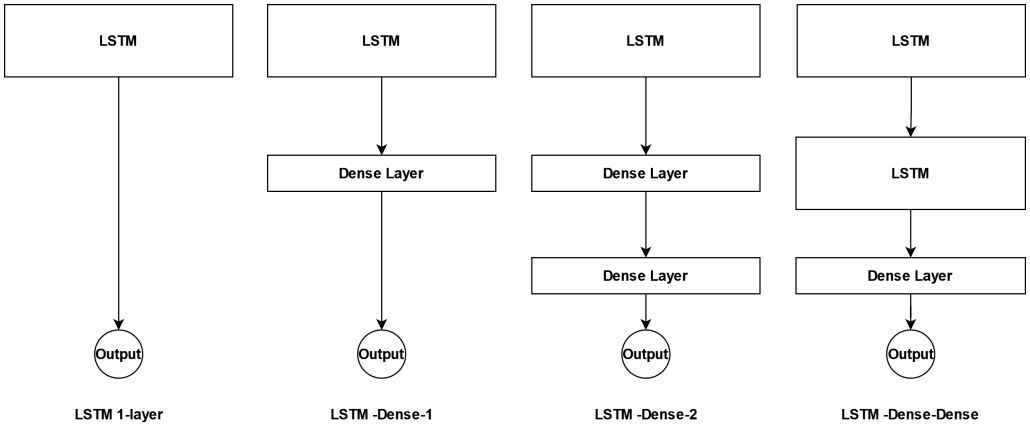

Figura 2. Esquema representativo das arquiteturas das LSTMs

Devido ao seu funcionamento focado em séries temporais, as entradas para as redes baseadas em LSTM precisavam ser sequências. Com isso, não bastava a entrada de uma amostra com a medição de cada sensor apenas, como de costume para todos os modelos utilizados até então. Nesse caso era necessária uma sequência de um tamanho definido para cada sensor como entrada. Isso exigiu um tratamento maior dos dados no caso desse tipo de arquitetura, culminando com uma entrada dessa arquitetura bem mais densa que a das outras redes, o que inviabilizou o uso de features polinomiais nesse tipo de modelo dada a exigência de memória para tanto.

Tabela 1. Hiperparâmetros utilizados

Hiperparâmetro	**Intervalo/Valores**
Tamanho da sequência	[30,100]
Clip	[80,140]
Grau do polinomial	[2,3]
Scaler	MinMax, Stardard
Quantidade de épocas	[1,50]
Validation Split (interno)	[0.1,0.9]
Tamanho do batch	[32,512]
Otimizador	Adam,RMSprop
Taxa de aprendizado	[1e-4, 1e-2]
LSTM Unidades	[16,512]
LSTM Ativador	tanh
LSTM Dropout	[0.1,0.9]
Dense Unidades	[16,512]
Dense Ativador	relu, elu, selu, tanh, sigmoid
Dense Dropout	[0.1,0.9]

Por fim, é importante destacar os hiperparâmetros dessas arquiteturas usados na sintonia, bem como seus intervalos e/ou valores usados. Essas informações estão destacadas na tabela 1.

3.3. Dataset

A primeira etapa do projeto foi a escolha do conjunto de dados a ser utilizado. Foi selecionado o dataset de motores Turbofan, montado pela NASA [Saxena et al. 2008]. Um esquema desse equipamento pode ser visto na figura 3, juntamente com um diagrama dos módulos internos e suas conexões na figura 4.

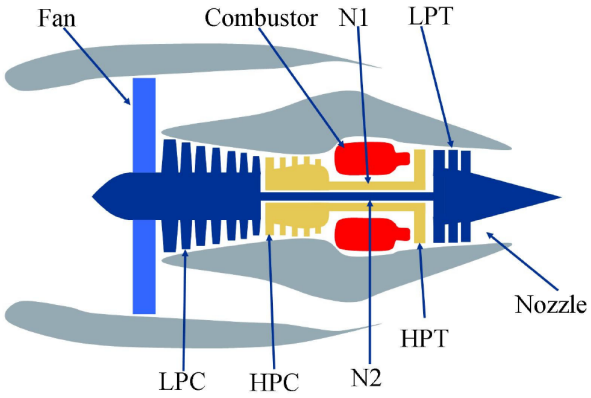

Figura 3. Diagrama simplificado do motor turbofan [Saxena et al. 2008]

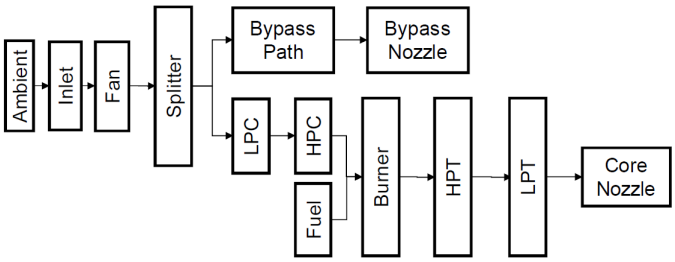

Figura 4. Esquema dos módulos do motor turbofan [Saxena et al. 2008]

Esses dados foram gerados a partir de simulações do *Commercial Modular Aero-Propulsion System Simulation* (C-MAPSS), uma ferramenta para a simulação realística de grandes motores turbofan comerciais, em que pode-se definir uma série de parâmetros de entrada para balizar os experimentos e obter vários valores como saída. No dataset, as saídas disponibilizadas são a de 21 sensores, listados na tabela 2, juntamente com três parâmetros que indicam a configuração daquela simulação com respeito à altitude de voo, número de Mach e temperatura.

Cada simulação é composta por séries temporais, uma para cada sensor, com uma medição por ciclo de operação. Cada motor começa num estado funcional, com algum grau de desgaste, mas que não afeta significativamente sua operação normal, até que, em algum momento da simulação, ocorre algum defeito ou uma falha, que aumenta até a falha completa do sistema, isto é, tem-se o run-to-failure. O objetivo é, portanto, estimar quantos ciclos de operação ainda restam para aquele motor, dados aquelas leituras dos sensores. Em outras palavras, a meta é estimar a RUL do motor, dadas as informações de seus sensores.

Tabela 2. Descrição dos sensores [Saxena et al. 2008]

#	Symbol	Description	Unit
0	T2	Total temperature at fan inlet	°R
1	T24	Total temperature at LPC outlet	°R
2	T30	Total temperature at HPC outlet	°R
3	T50	Total temperature at LPT outlet	°R
4	P2	Pressure at fan inlet	psia
5	P15	Total pressure in bypass-duct	psia
6	P30	Total pressure at HPC outlet	psia
7	Nf	Physical fan speed	rpm
8	Nc	Physical core speed	rpm
9	epr	Engine pressure ratio (P50/P2)	–
10	Ps30	Static pressure at HPC outlet	psia
11	phi	Ratio of fuel flow to Ps30	pps/psi
12	NRf	Corrected fan speed	rpm
13	NRc	Corrected core speed	rpm
14	BPR	Bypass Ratio	–
15	farB	Burner fuel-air ratio	–
16	htBleed	Bleed Enthalpy	–
17	Nf_dmd	Demanded fan speed	rpm
18	PCNfR_dmd	Demanded corrected fan speed	rpm
19	W31	HPT coolant bleed	lbm/s
20	W32	LPT coolant bleed	lbm/s

Tabela 3. Especificações dos grupos do dataset

Grupo de Dados	FD001	FD002	FD003	FD004
Simulações Treino	100	260	100	248
Simulações Teste	100	259	100	249
Condições de Simulação	1	6	1	6
Tipos de Falhas	1	1	2	2

O dataset é dividido em quatro grupos, com diferentes graus de complexidade para essa tarefa. Os dados vêm previamente separados em treino e em teste para a elaboração dos modelos. Como é possível ver na tabela 3, o grupo FD001 é o mais simples, ao passo que o FD004 é o mais complexo. No FD001 e FD003, a única condição de simulação é no nível do mar. Nos outros dois, são seis diferentes condições. No FD001 e FD002, há apenas um tipo de defeito, causado por degradação no Compressor de Alta Pressão (HPC, High-Pressure Compressor). Nos outros dois, há, além desse possível defeito, também degradação na ventoinha (fan) do motor.

3.4. Cálculo da RUL

No dataset, somente há anotações de RUL corretas no conjunto de teste. Por isso, para o treino, é preciso fazer um processamento inicial das leituras dos sensores. Inicialmente, o intuito era calcular a RUL de forma linear, ou seja, a cada ciclo, ela diminui em uma unidade. Tendo em vista que as simulações do treino são run-to-failure, pode-se definir que

o último ciclo de cada observação tem RUL 0 e incrementar os antecessores linearmente, como descrito.

Porém, como ressaltam [Berghout and Benbouzid 2022, Saxena et al. 2008], nessas simulações de treino, os motores apresentam leituras de forma que eles tendem a manter o seu funcionamento até que, em algum momento, ocorre uma falha, ou um princípio de falha, e, a partir desse instante, se nada for feito o sistema apresentará instabilidades graves. Assim, há uma não-linearidade nesse valor, ou seja, de início, não é possível saber quando haverá um problema, apenas quando surgirem princípios de um defeito pode-se estimar um valor.

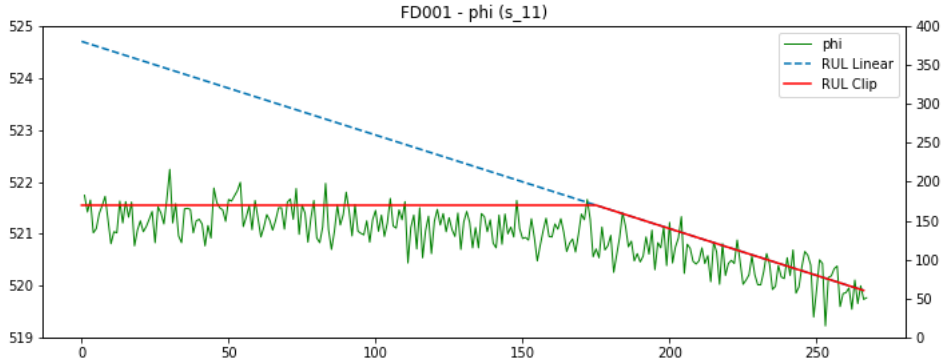

Figura 5. Ilustração da não-linearidade

Por exemplo, na figura 5, nota-se a leitura de um sensor para uma simulação do grupo FD001. Do início das leituras até o aproximadamente o ciclo 270, seu valor fica num patamar em torno de 521.5. Após esse instante, com o surgimento do defeito, é possível estimar quantos ciclos até a falha. Porém, antes disso, uma RUL totalmente linear não capturaria esse comportamento de continuidade de funcionamento até o a aparecimento da irregularidade.

Na primeira etapa, em termos de modelagem, essa não-linearidade foi implementada como mais um parâmetro dos modelos, a ser sintonizado com o refino, por meio da função *clip* da biblioteca Numpy. Ela era feita logo antes do treino do modelo, sendo aplicada no vetor de resposta (y). Assim, por exemplo, um vetor com o comportamento da linha tracejada do gráfico 5 era transformado na linha vermelha contínua através da função *clip*.

Já nesse segundo momento, como comentado anteriormente, foram criadas classes de modelos customizados do scikit-learn que serviam de invólucro (*wrapper*) dos modelos do Tensorflow-Keras. Isso permitiu que o pré-processamento fosse feito de forma mais personalizada, o que incluía a estandardização dos dados de entrada, uma eventual adição de features polinomiais e clipagem do vetor de resposta no valor requisitado, agora usando o *clip* para dataframes Pandas. Do ponto de vista prático, no entanto, o efeito era o mesmo do anterior, mas de uma forma mais orientada a objeto, com uma classe dedicada para todas as operações que concernem um modelo.

4. Resultados e Discussão

O desempenho de cada modelo nos conjuntos de teste pode ser observado nas tabelas a seguir. Na tabela 4, vê-se a raiz do erro quadrado médio, enquanto na 5, é possível ver o coeficiente de determinação R^2. Estão dispostos apenas os dados dos modelos que usaram a RUL não-linear, tendo em vista que os com RUL linear tiveram um desempenho significativamente pior em todos os casos. Ainda assim, os resultados com esses outros modelos podem ser obtidos do apêndice A.

Nas tabelas apresentadas, também constam dois outros resultados: o do Dummy Regressor para servir de desempenho-base e o RF, destaque da etapa anterior, com intuito de se ter uma comparação com os modelos tradicionais. No apêndice B, é possível encontrar os resultados completos obtidos na primeira fase.

Tabela 4. RMSE, usando RUL não-linear em redes neurais

Configuração	Modelo	**FD001**	**FD002**	**FD003**	**FD004**
	Dummy Regressor	45.115	53.834	44.158	60.690
	RF	12.761	12.769	13.237	13.767
	ANN 1-layer	10.152	11.565	9.917	11.002
Base	ANN 2-layer	9.515	12.159	11.887	11.318
Features	ANN 3-layer	9.612	11.148	10.424	11.799
	ANN 4-layer	10.118	11.215	12.221	23.510
	LSTM 1-layer	12.909	19.010	5.472	23.884
	LSTM-Dense-1	14.201	13.821	9.985	24.343
	LSTM-Dense-2	17.624	11.743	7.554	20.492
	LSTM-LSTM-Dense	8.219	22.862	9.882	19.677
	Dummy Regressor	44.864	53.879	43.915	60.381
	RF	13.609	10.122	15.112	10.943
Polynomial	ANN 1-layer	9.428	14.426	14.569	15.395
Features	ANN 2-layer	11.927	14.709	10.379	19.508
	ANN 3-layer	13.132	13.710	12.086	12.765
	ANN 4-layer	10.585	17.804	22.091	23.119

A primeira observação a ser feita é similar ao contexto do primeiro estágio, no que concerne a RUL linear. Em outras palavras, se mantém a máxima de ML e da computação em geral: ”garbage in, garbage out”, em que dados mal tratados levam a resultados imprecisos, ou mesmo inconsistentes. No caso de ML, nem mesmo bons algoritmos e arquiteturas de redes neurais, com uma excelente sintonia de hiperparâmetros conseguirão superar uma entrada com problemas.

Também em consonância com as reflexões da primeira etapa, há de se ressaltar o horizonte de tempo necessário para o treinamento dos modelos. Naturalmente, a quantidade de épocas definida tinha uma alta influência, mas, além desse fator, é necessário evidenciar o tipo de arquitetura. As LSTMs, muito por conta de seu funcionamento de RNN, eram mais complexas e exigem mais poder computacional. Ademais, modelos com mais camadas e, principalmente, com mais layers de LSTM, eram bem mais demorados de se treinar.

Com base nos resultados apresentados nas tabelas 4 e 5, é notável que não há uma grande homogeneidade de resultados tal como se viu no aprendizado tradicional,

Tabela 5. R^2, usando RUL não-linear em redes neurais

Configuração	Modelo	FD001	FD002	FD003	FD004
Base Features	Dummy Regressor	-0.179	-0.002	-0.138	-0.239
	RF	0.839	0.839	0.812	0.803
	ANN 1-layer	0.855	0.829	0.845	0.834
	ANN 2-layer	0.873	0.811	0.777	0.825
	ANN 3-layer	0.870	0.841	0.829	0.810
	ANN 4-layer	0.856	0.839	0.817	0.701
	LSTM 1-layer	0.869	0.835	0.954	0.725
	LSTM-Dense-1	0.883	0.895	0.856	0.538
	LSTM-Dense-2	0.724	0.901	0.928	0.756
	LSTM-LSTM-Dense	0.944	0.582	0.913	0.775
Polynomial Features	Dummy Regressor	-0.166	-0.004	-0.125	-0.226
	RF	0.838	0.857	0.776	0.820
	ANN 1-layer	0.875	0.734	0.814	0.771
	ANN 2-layer	0.800	0.770	0.830	0.573
	ANN 3-layer	0.758	0.785	0.825	0.822
	ANN 4-layer	0.843	0.707	0.316	0.718

sobretudo em redes mais complexas. As arquiteturas que utilizavam camadas de LSTM tinham problemas de generalização em cenários com várias condições de operação, ou seja, no FD002 e FD004. Por outro lado, nos outros dois cenários chegou a apresentar os melhores resultados obtidos em todo o projeto, com RMSE abaixo de 10 e R^2 acima de 0.9. Em outro âmbito, as ANNs se mostraram bastante regulares entre os vários cenários, sendo melhores mesmo que os modelos RF da primeira etapa.

Fazendo uma avaliação da adição de features polinomiais, nota-se que, em geral, o efeito não foi positivo, diferentemente do observado no aprendizado supervisionado tradicional. Tanto a RMSE quanto o R^2 ou se mantiveram em patamares similares ou pioraram levemente com a inserção desses dados. Em alguns poucos casos, há uma leve melhora da performance, portanto, nesses momentos há de se levar me conta o tradeoff entre complexidade, tempo de treino e desempenho.

Num panorama geral, nota-se que os resultados da ANN com três camadas foram os mais consistentes entre os vários cenários, sem apresentar grandes diferenças para as situações mais complexas do cenário FD004, em que várias arquiteturas tiveram dificuldades. No contexto daquelas baseadas em LSTM, convém destacar a com uma simples única camada, que apesar de uma performance aquém se comparada às ANNs, conseguiu se sobressair perante as RNNs.

Finalmente, é interessante comparar os resultados aqui obtidos com outros estudos e notar que há uma semelhança. Nas redes propostas por [Li et al. 2018] para o cenário FD001, tem-se uma RMSE de 13, o que está próximo aos valores obtidos aqui, com as diferenças nas decisões tomadas para alguns hiperparâmetros-chave como número de épocas, valores de clip e quantidade de camadas. Em [Mathew et al. 2017], apesar de não ficar explícito qual arquitetura exatamente de DL foi usada, observa-se uma RMSE em torno de 40, bastante díspar com os aqui encontrados, mas compreensível devido a diferentes escolhas na sintonia ou no treinamento dos dados.

4.1. Análise de Valores SHAP

Tal como anteriormente, é importante incrementar a explicabilidade e a transparência das estimativas feitas. Isso será feito por meior da análise dos valores SHAP para os modelos treinados a partir da biblioteca de mesmo nome. Haverá um foco apenas nos modelos ANN de 3 camadas, tendo em vista que tiveram bons resultados bastante homogêneos nos vários cenários propostos. Com o intuito de analisar arquiteturas de redes neurais e similares, a biblioteca traz métodos otimizados para esse tipo de modelo, o que agiliza e facilita o cômputo dos valores SHAP e suas análises.

Ademais, também não serão averiguados os modelos com o uso de features polinomiais, tendo em vista que há uma divisão excessiva de importância dos atributos, o que leva a uma granularização dos valores SHAP entre as inúmeras features, as deixando com importância baixa, tal como observado na figura 6. Isso também ocorreu com os modelos de aprendizado supervisionado, apresentando um tradeoff entre um melhor desempenho, proporcionado àquele momento com o uso de tais estratégias, e maior explicabilidade. No caso das redes neurais estudadas, todavia, como não há um uma melhoria de perfomance significativa com o essa abordagem, não há tradeoff, apenas mais um motivo que advoga em desfavor ao uso de tais features no contexto de redes neurais para esses dados.

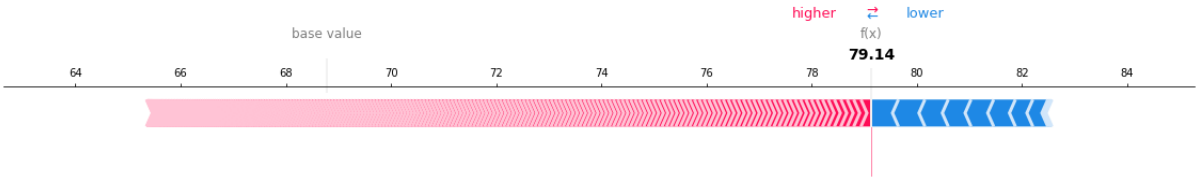

Figura 6. Valores SHAP de uma amostra com Polynomial Features

Nesse momento, serão feitas análises dos valores SHAP para cada cenário. Os gráficos para a importância de cada atributo para o modelo RF, os quais foram apresentados na etapa anterior do projeto, estão no apêndice C para ulterior comparação, caso necessário.

No FD001, na figura 7, nota-se uma maior importância aos atributos relacionados diretamente ao compressor HPC, como o Ps30, o P30 e o phi, mas também com módulos após ele, dispostos no diagrama funcional, figura 4, como T50, do LPT, e as velocidades do núcleo, Nc e NRc. Assim, há uma prioridade de features relacionadas direta ou indiretamente a alguma alteração no funcionamento do HPC, exatamente o defeito das simulações deste conjunto de dados.

Nota-se uma diferença de importância relativa entre o modelo RF e o ANN de 3 camadas. Contudo, é interessante notar que, em termos de qual o impacto de cada feature no valor da RUL, as conclusões são bastantes similar. Por exemplo, o Ps30 tende pouco a aumentar levemente a RUL em poucas unidades (agregado em azul à esquerda do eixo central), mas favorece bastante a diminuição dela em até cerca de 10 unidades (amostras vermelhas à esquerda).

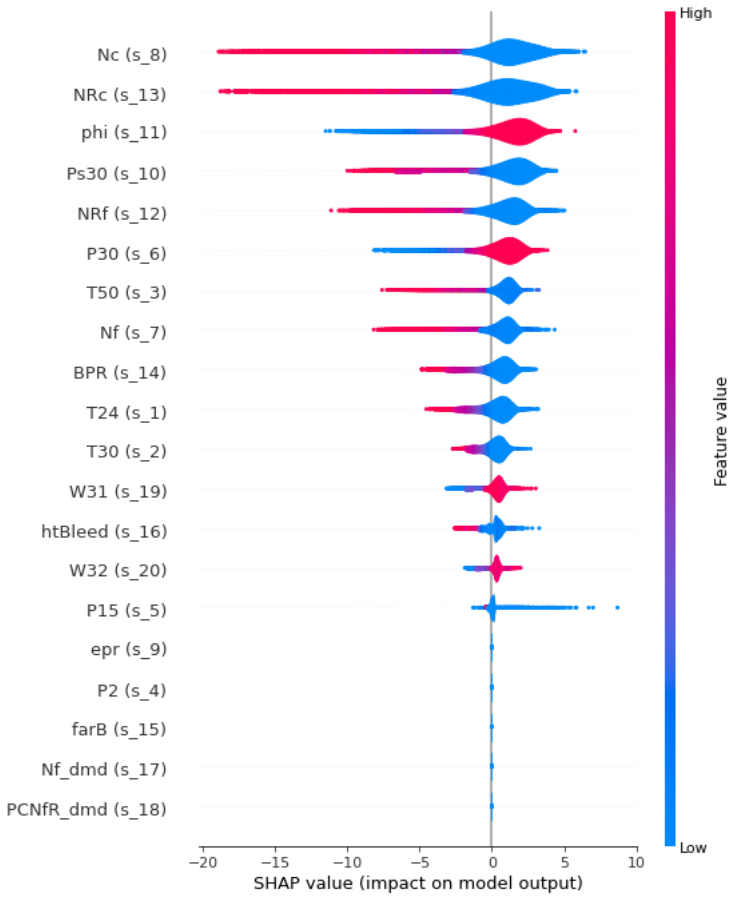

Figura 7. Valores SHAP para o grupo FD001

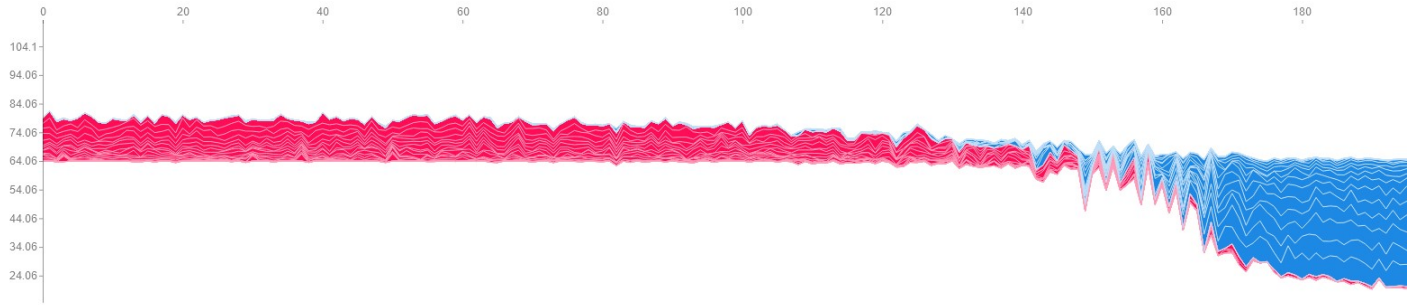

Figura 8. Evolução de valores SHAP da simulação 42 do FD001

Figura 9. Média móvel da evolução de valores SHAP da simulação 42 do FD001

Pode-se exemplificar esse comportamento com uma simulação feita, retratada na figura 8. Ao longo da simulação, é possível ver os parâmetros que contribuem para o aumento da estimativa da RUL, coloridos em tom avermelhado, a partir do valor médio, bem como os que contribuem para sua diminuição, coloridos em azul. O valor da estimativa está no limiar entre a área azul e a área avermelhada, de forma que sempre a azul fica por cima e a vermelha por baixo. Com isso, é possível perceber que, aproximadamente no ciclo 140, o modelo passa a detectar o defeito e o motor apresenta instabilidades cada vez maiores (área azul), até o fim da simulação.

Usando os valores sem qualquer tratamento, na figura 8, nota-se bastantes oscilações, mas, na média, essas variações são pequenas. Assim, aplicou-se uma média móvel nos valores SHAP calculados, o que proporciona uma evolução muito mais suave, disposta na figura 9. Num uso prático, é pertinente considerar um tratamento dessa natureza antes de se utilizar essa saída de dados, a fim de minimizar ruídos e oscilações, advindas, inclusive, das entradas.

Figura 10. Valores SHAP da simulação 42 do FD001

Fazendo um recorte nesse gráfico, é possível descobrir quais atributos contribuem em cada situação, na figura 10. Nota-se que os principais parâmetros elencados na figura 7 são exemplificados aqui, sendo que esses atributos, muito relacionados ao HPC e seus efeitos, inicialmente ajudam a denotar a estabilidade do sistema, mas, posteriormente, indicam o defeito e, por isso, a diminuição da RUL.

Análises semelhantes podem ser feitas para os outros grupos de dados. No grupo FD002, retratado na figura 11, em que também há apenas um defeito, mas são feitas diferentes configurações de simulação, é esperado que os atributos ligados ao HPC, como Ps30 e T50 continuem importantes, o que de fato ocorreu. Apesar disso, outros parâmetros como htBleed e BPR ganharam importância. Convém destacar que atributos de configuração não foram considerados muito relevantes para o modelo tomar suas decisões, não figurando, portanto, entre as dez features mais importantes. Comparando com os resultados no RF, tem-se também um comportamento similar.

Ao se averiguar a evolução dos valores SHAP de uma simulação, na figura 12, nota-se uma maior instabilidade das estimativas, mesmo durante a fase inicial da experiência. Esse fenômeno também foi observado nas análises do aprendizado supervisionado, contudo o efeito aqui foi amplificado. Nota-se que algumas features têm influência de mais de 100 unidades para uma amostra. Com isso, a escala do gráfico dificulta a real noção de como se comporta a RUL, que está exatamente no limiar das áreas azul e rosa.

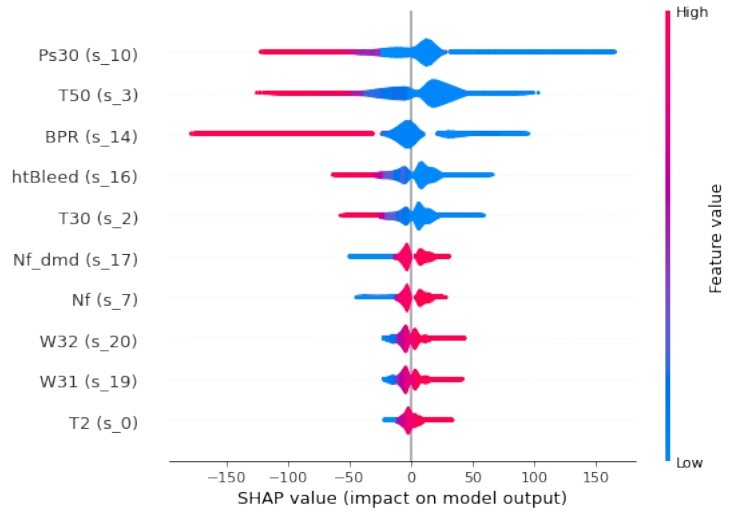

Figura 11. Dez maiores valores SHAP para o grupo FD002

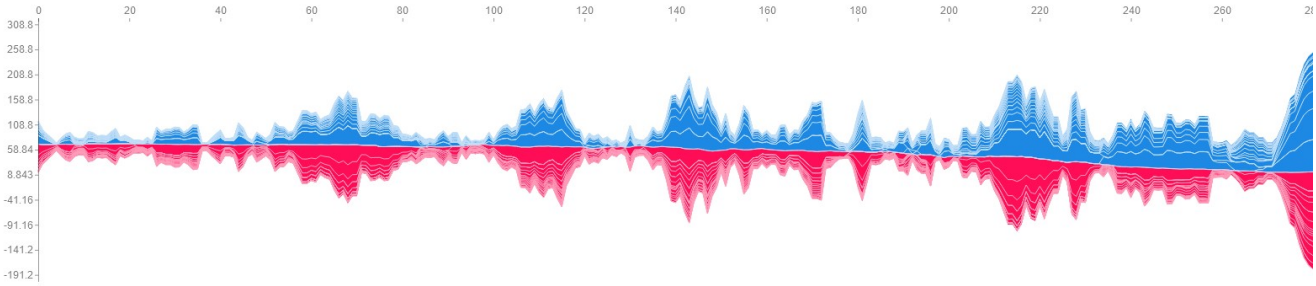

Figura 12. Média móvel da evolução de valores SHAP da simulação 32 do FD002

Tendo em vista que esse acontecimento também foi observado na primeira etapa, pode-se pensar que algumas das causas sejam similares, como ao aumento de importância de atributos que não deveriam obtê-lo, mas devido às diferentes configurações infundiu-se fatores de confusão para o modelo. É possível notar, numa análise do gráfico, que o valor da RUL diminui de fato. Isso é ressaltado também pela investigação das amostras em si, em contraste com toda a evolução, como visto na figura 13.

Nos dois últimos grupos, é possível fazer uma outra análise. Como agora são dois tipos de defeitos que podem ocorrer nos motores durante a simulação, e, nos dois primeiros datasets, o modelo usou os atributos ligados ao defeito para fazer a estimativa, espera-se que ele use, agora, também parâmetros ligados ao segundo tipo de falha para suas previsões.

Tal qual ressaltado previamente em [Saxena et al. 2008], o outro tipo de problema possível está ligado a uma degradação da ventoinha e, conforme as expectativas, atributos relacionados à ela, como Nf, NRc, NRf e T2 ganham destaque e, juntamente com eles, os já citados ligados ao HPC continuam importantes para os modelos, como denotam as figuras 14 e 15. Convém destacar que, também no FD004, tal como no FD002, a evolução dos valores SHAP apresenta magnitudes bastante elevadas, novamente distorcendo os gráficos. Contudo, ao analisar as amostras em si, é possível notar que, de fato, a RUL diminui ao longo do tempo, de forma similar ao que ocorre no FD002 na figura 13.

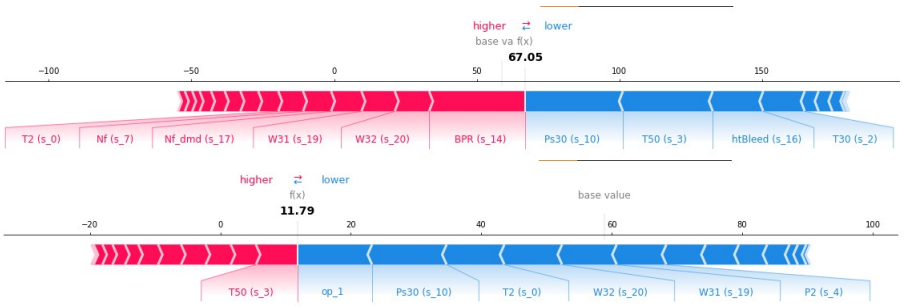

Figura 13. Valores SHAP da simulação 42 do FD002

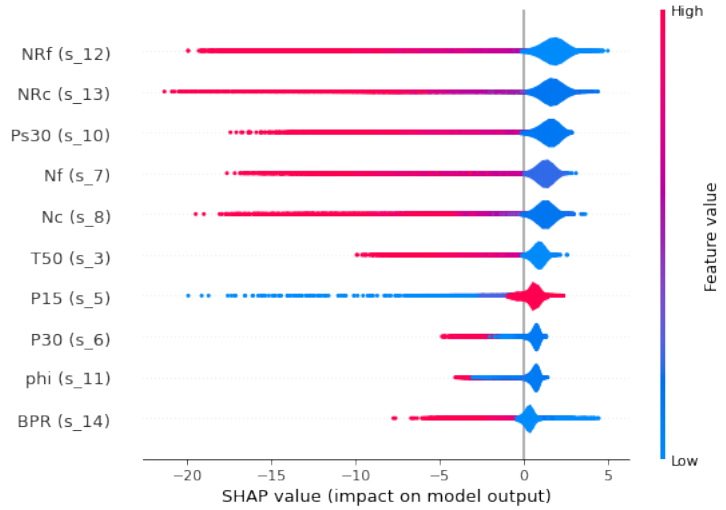

Figura 14. Dez maiores valores SHAP para o grupo FD003

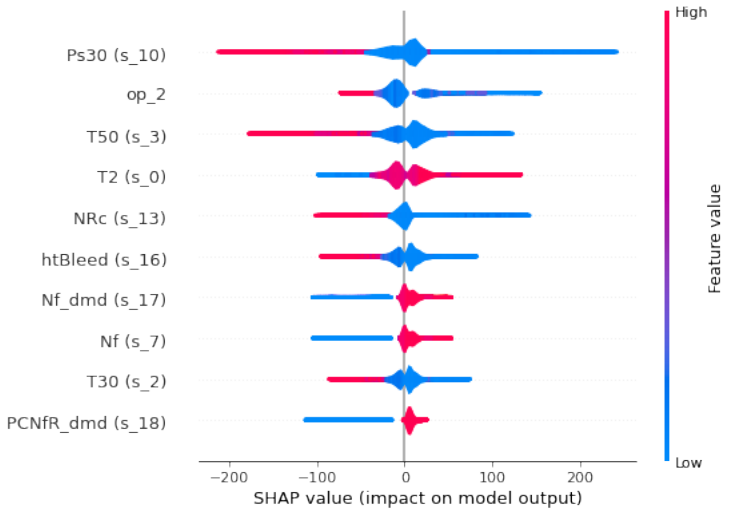

Figura 15. Dez maiores valores SHAP para o grupo FD004

Com os dos defeitos existentes, é possível encontrar tanto simulações que apresentaram o defeito no HPC, com um comportamento semelhante à figura 10, como também aquelas com defeito na ventoinha, como visto na figura 16. Nesse caso, na simulação 42 do FD003, o parâmetro NRf foi bastante importante para a diminuição da RUL.

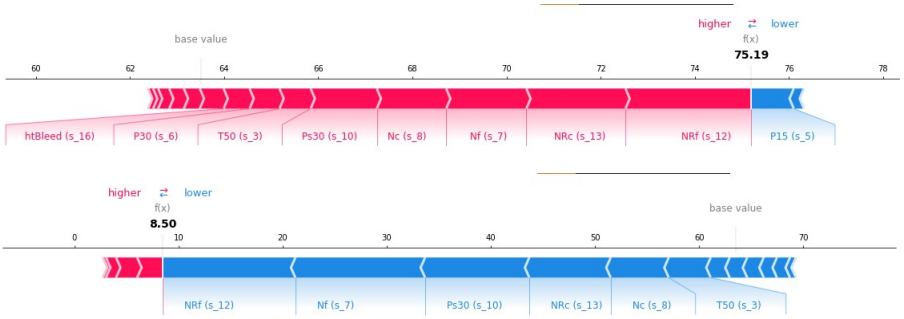

Figura 16. Valores SHAP da simulação 42 do FD003

Desse modo, por meio da análise dos valores SHAP dos modelos, é possível evidenciar a importância dada aos diversos atributos, relacionando-os com os efeitos que geraram a base dados. Isso permite, por exemplo, identificar o momento em que o motor começou a apresentar o defeito, bem como quais sensores contribuíram para a estimativa de RUL feita pelo modelo. Analogamente, pode-se descobrir quais são os parâmetros que não tem relevância para essa tarefa. Além disso, os valores SHAP permitem quantificar essa importância para as amostras de entrada, com mais profundidade que apenas os vetores de coeficientes ou de importância de features próprios dos modelos em si.

Essa análise, na esteira das recentes discussões de ML acerca da explicabilidade de modelos, permite que eles não sejam tanto um método ”caixa-preta”. Em outras palavras, portanto, garante-se mais transparência e interpretabilidade aos resultados e previsões feitas. Dessa forma, tem-se mais confiabilidade mesmo de modelos mais complexos como os de redes neurais.

5. Conclusão

Ao final deste projeto, foi possível averiguar o desempenho de diversos modelos de aprendizado supervisionado, de arquiteturas de redes neurais artificiais e de redes recorrentes para a tarefa da previsão da RUL a partir de dados de sensores de motores turbofan.

Após a análise de diferentes estratégias e configurações, constatou-se que os modelos baseados em árvore apresentaram um melhor desempenho em todos os cenários estudados, com um satisfatório grau de generalização. No campo das redes neurais, apesar de não se ter uma grande homogeneidade nos vários cenários, as ANNs com 3 camadas densas se mostraram as mais regulares nas várias situações. Não obstante, convém destacar as redes baseadas em LSTM, as quais, em casos específicos, obtiveram os melhores resultados dentre todos os modelos investigados nas duas etapas.

Além disso, por meio da análise de valores SHAP, foi possível interpretar e explicar as estimativas feitas pelos modelos nos vários cenários, garantindo mais transparência e explicabilidade para os resultados.

Como possibilidades de investigação futura, pode-se selecionar uma outra base de dados para averiguar esses e outros modelos, a fim de se verificar as conclusões aqui salientadas. Pode-se também elaborar outras arquiteturas, sejam elas mais profundas, seja utilizando outras metodologias como redes convolucionais ou outras estratégias de processamento como suavização exponencial. Outra perspectiva está em buscar uma oportunidade de aplicar esses modelos num contexto experimental real, possibilitando um auxílio no planejamento de atividades de manutenção de uma indústria, por exemplo.

Finalmente, é importante destacar que o projeto permitiu a aplicação de conhecimentos de ML, Ciência dos Dados, Redes Neurais e DL, bem como diversas outras áreas da Ciência da Computação, com as quais houve contato ao longo da graduação. Além disso, foi possível vivenciar um fluxo de trabalho de aprendizado de máquina, incluindo a criação de modelos, com suas inerentes complexidades e particularidades, bem como implementar soluções para o tratamento inicial dos dados, encapsulamento de arquiteturas, para, ao final, com os modelos já treinados, ponderar acerca dos resultados de forma abalizada.

Referências

[Angelopoulos et al. 2020] Angelopoulos, A., Michailidis, E. T., Nomikos, N., Trakadas, P., Hatziefremidis, A., Voliotis, S., and Zahariadis, T. (2020). Tackling faults in the industry 4.0 era—a survey of machine-learning solutions and key aspects. *Sensors*, 20(1).

[Berghout and Benbouzid 2022] Berghout, T. and Benbouzid, M. (2022). A systematic guide for predicting remaining useful life with machine learning. *Electronics*, 11(7).

[Carvalho et al. 2019] Carvalho, T. P., Soares, F. A., Vita, R., Francisco, R. d. P., Basto, J. P., and Alcalá, S. G. (2019). A systematic literature review of machine learning methods applied to predictive maintenance. *Computers & Industrial Engineering*, 137:106024.

[Chollet et al. 2015] Chollet, F. et al. (2015). Keras. `https://keras.io`.

[Ferreira and Gonçalves 2022] Ferreira, C. and Gonçalves, G. (2022). Remaining useful life prediction and challenges: A literature review on the use of machine learning methods. *Journal of Manufacturing Systems*, 63:550–562.

[Harris et al. 2020] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

[Head et al. 2020] Head, T., Kumar, M., Nahrstaedt, H., Louppe, G., and Shcherbatyi, I. (2020). scikit-optimize/scikit-optimize.

[Hunter 2007] Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

[Jagatheesaperumal et al. 2021] Jagatheesaperumal, S. K., Rahouti, M., Ahmad, K., Al-Fuqaha, A., and Guizani, M. (2021). The duo of artificial intelligence and big data

for industry 4.0: Applications, techniques, challenges, and future research directions. *IEEE Internet of Things Journal*, pages 1–1.

[Leser 2017] Leser, P. E. (2017). *Probabilistic Prognostics and Health Management for Fatigue-Critical Components using High-Fidelity Models*. PhD thesis, North Carolina State University.

[Li et al. 2018] Li, X., Ding, Q., and Sun, J.-Q. (2018). Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering System Safety*, 172:1–11.

[Lundberg and Lee 2017] Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc.

[Mathew et al. 2017] Mathew, V., Toby, T., Singh, V., Rao, B. M., and Kumar, M. G. (2017). Prediction of remaining useful lifetime (rul) of turbofan engine using machine learning. In *2017 IEEE International Conference on Circuits and Systems (ICCS)*, pages 306–311.

[Pedregosa et al. 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Peres et al. 2020] Peres, R. S., Jia, X., Lee, J., Sun, K., Colombo, A. W., and Barata, J. (2020). Industrial artificial intelligence in industry 4.0 - systematic review, challenges and outlook. *IEEE Access*, 8:220121–220139.

[Prytz et al. 2015] Prytz, R., Nowaczyk, S., Rögnvaldsson, T., and Byttner, S. (2015). Predicting the need for vehicle compressor repairs using maintenance records and logged vehicle data. *Engineering Applications of Artificial Intelligence*, 41:139–150.

[Russell and Norvig 2020] Russell, S. J. and Norvig, P. (2020). *Artificial Intelligence: A Modern Approach*. Pearson, Englewood Cliffs, N.J.

[Saxena et al. 2008] Saxena, A., Goebel, K., Simon, D., and Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 International Conference on Prognostics and Health Management*, pages 1–9.

[Serradilla et al. 2022] Serradilla, O., Zugasti, E., Rodriguez, J., and Zurutuza, U. (2022). Deep learning models for predictive maintenance: a survey, comparison, challenges and prospects. *Applied Intelligence*, 52(10):10934–10964.

[Silvestrin et al. 2019] Silvestrin, L. P., Hoogendoorn, M., and Koole, G. (2019). A comparative study of state-of-the-art machine learning algorithms for predictive maintenance. In *SSCI*, pages 760–767.

[Susto et al. 2015] Susto, G. A., Schirru, A., Pampuri, S., McLoone, S., and Beghi, A. (2015). Machine learning for predictive maintenance: A multiple classifier approach. *IEEE Transactions on Industrial Informatics*, 11(3):812–820.

[TensorFlow Developers 2022] TensorFlow Developers (2022). Tensorflow.

[Wang et al. 2020] Wang, Y., Zhao, Y., and Addepalli, S. (2020). Remaining useful life prediction using deep learning approaches: A review. *Procedia Manufacturing*, 49:81–88. Proceedings of the 8th International Conference on Through-Life Engineering Services – TESConf 2019.

[Wes McKinney 2010] Wes McKinney (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61.

[Wu et al. 2019] Wu, J., Chen, X.-Y., Zhang, H., Xiong, L.-D., Lei, H., and Deng, S.-H. (2019). Hyperparameter optimization for machine learning models based on bayesian optimizationb. *Journal of Electronic Science and Technology*, 17(1):26–40.

[Yuan et al. 2016] Yuan, M., Wu, Y., and Lin, L. (2016). Fault diagnosis and remaining useful life estimation of aero engine using lstm neural network. In *2016 IEEE International Conference on Aircraft Utility Systems (AUS)*, pages 135–140.

[Zonta et al. 2020] Zonta, T., da Costa, C. A., da Rosa Righi, R., de Lima, M. J., da Trindade, E. S., and Li, G. P. (2020). Predictive maintenance in the industry 4.0: A systematic literature review. *Computers Industrial Engineering*, 150:106889.

A. Resultados Completos em Redes Neurais

Tabela 6. R^2, usando RUL não-linear em redes neurais

Configuração	Modelo	FD001	FD002	FD003	FD004
	Dummy Regressor	-0.179	-0.002	-0.138	-0.239
	RF	0.839	0.839	0.812	0.803
	ANN 1-layer	0.855	0.829	0.845	0.834
Base	ANN 2-layer	0.873	0.811	0.777	0.825
Features	ANN 3-layer	0.870	0.841	0.829	0.810
	ANN 4-layer	0.856	0.839	0.817	0.701
	LSTM 1-layer	0.869	0.835	0.954	0.725
	LSTM-Dense-1	0.883	0.895	0.856	0.538
	LSTM-Dense-2	0.724	0.901	0.928	0.756
	LSTM-LSTM-Dense	0.944	0.582	0.913	0.775
	Dummy Regressor	-0.166	-0.004	-0.125	-0.226
	RF	0.838	0.857	0.776	0.820
Polynomial	ANN 1-layer	0.875	0.734	0.814	0.771
Features	ANN 2-layer	0.800	0.770	0.830	0.573
	ANN 3-layer	0.758	0.785	0.825	0.822
	ANN 4-layer	0.843	0.707	0.316	0.718

Tabela 7. RMSE, usando RUL não-linear em redes neurais

Configuração	Modelo	**FD001**	**FD002**	**FD003**	**FD004**
	Dummy Regressor	45.115	53.834	44.158	60.690
	RF	12.761	12.769	13.237	13.767
	ANN 1-layer	10.152	11.565	9.917	11.002
Base	ANN 2-layer	9.515	12.159	11.887	11.318
Features	ANN 3-layer	9.612	11.148	10.424	11.799
	ANN 4-layer	10.118	11.215	12.221	23.510
	LSTM 1-layer	12.909	19.010	5.472	23.884
	LSTM-Dense-1	14.201	13.821	9.985	24.343
	LSTM-Dense-2	17.624	11.743	7.554	20.492
	LSTM-LSTM-Dense	8.219	22.862	9.882	19.677
	Dummy Regressor	44.864	53.879	43.915	60.381
	RF	13.609	10.122	15.112	10.943
Polynomial	ANN 1-layer	9.428	14.426	14.569	15.395
Features	ANN 2-layer	11.927	14.709	10.379	19.508
	ANN 3-layer	13.132	13.710	12.086	12.765
	ANN 4-layer	10.585	17.804	22.091	23.119

Tabela 8. R^2, usando RUL linear em redes neurais

Configuração	Modelo	FD001	FD002	FD003	FD004
	Dummy Regressor	-0.604	-0.251	-2.298	-0.735
	RF	0.462	0.695	-0.311	0.424
	ANN 1-layer	0.506	0.639	-0.477	0.154
Base	ANN 2-layer	0.466	0.613	-1.014	0.244
Features	ANN 3-layer	0.442	0.639	-0.641	-0.243
	ANN 4-layer	0.610	0.640	-0.180	0.294
	LSTM 1-layer	0.807	0.625	0.611	0.365
	LSTM-Dense-1	0.793	0.730	0.550	0.504
	LSTM-Dense-2	0.767	0.754	0.398	-0.056
	LSTM-LSTM-Dense	0.708	0.488	-0.111	0.278
	Dummy Regressor	-0.604	-0.251	-2.298	-0.735
	RF	0.422	0.692	-0.314	0.446
Polynomial	ANN 1-layer	0.597	0.669	-0.375	0.320
Features	ANN 2-layer	0.210	0.623	-0.706	0.241
	ANN 3-layer	0.416	0.591	-0.398	-0.848
	ANN 4-layer	0.177	0.588	0.266	0.117

Tabela 9. RMSE, usando RUL linear em redes neurais

Configuração	Modelo	FD001	FD002	FD003	FD004
	Dummy Regressor	52.625	60.162	75.180	71.828
	RF	30.481	29.683	47.403	41.397
	ANN 1-layer	29.199	32.310	50.311	50.149
Base	ANN 2-layer	30.355	33.435	58.744	47.413
Features	ANN 3-layer	31.051	32.315	53.034	60.796
	ANN 4-layer	25.966	32.261	44.970	45.809
	LSTM 1-layer	18.271	32.923	25.813	43.447
	LSTM-Dense-1	18.928	27.967	27.754	38.404
	LSTM-Dense-2	20.064	26.654	32.126	56.017
	LSTM-LSTM-Dense	22.445	38.494	43.638	46.342
	Dummy Regressor	52.625	60.162	75.180	71.828
	RF	31.601	29.844	47.460	40.577
Polynomial	ANN 1-layer	26.379	30.927	48.532	44.954
Features	ANN 2-layer	36.946	33.018	54.072	47.491
	ANN 3-layer	31.766	34.394	48.943	74.113
	ANN 4-layer	37.689	34.518	35.461	51.246

B. Resultados Completos com Aprendizado Supervisionado Tradicional

Tabela 10. R^2, usando RUL não-linear com métodos tradicionais

Configuração	Modelo	FD001	FD002	FD003	FD004
Base Features	Dummy Regressor	-0,179	-0,002	-0,138	-0,239
	kNN	0,530	0,742	0,761	0,770
	Linear Regressor	0,623	0,767	0,720	0,687
	SGD Regressor	0,676	0,764	0,755	0,647
	Linear SVR	0,614	0,766*	0,744	0,682*
	DT	0,819	0,800	0,756	0,742
	RF	0,839	0,839	0,812	0,803
Polynomial Features	Dummy Regressor	-0,166	-0,004	-0,125	-0,226
	kNN	0,720	0,812	0,780	0,793
	Linear Regressor	0,596	0,766	0,753	0,686
	SGD Regressor	0,715	0,761	0,733	0,618
	Linear SVR	0,789	0,774*	0,764	-4,017**
	DT	0,711	0,794	0,776	0,768
	RF	0,838	0,857	0,776	0,820

*Teve problemas de convergência
** Não convergiu

Tabela 11. RMSE, usando RUL não-linear com métodos tradicionais

Configuração	Modelo	FD001	FD002	FD003	FD004
Base Features	Dummy Regressor	45,115	53,834	44,158	60,690
	kNN	28,477	15,957	16,215	14,690
	Linear Regressor	25,526	17,697	20,287	20,646
	SGD Regressor	23,645	18,286	17,036	-27,183
	Linear SVR	25,807	16,424*	18,062	21,377*
	DT	14,227	15,532	15,577	17,293
	RF	12,761	12,769	13,237	13,767
Polynomial Features	Dummy Regressor	44,864	53,879	43,915	60,381
	kNN	21,998	11,595	11,266	16,763
	Linear Regressor	26,415	16,445	17,117	21,234
	SGD Regressor	22,171	13,885	19,049	21,902
	Linear SVR	14,641	15,647*	14,513	-98,189**
	DT	12,322	12,161	14,454	15,868
	RF	13,609	10,122	15,112	10,943

*Teve problemas de convergência
** Não convergiu

Tabela 12. R^2, usando RUL linear com métodos tradicionais

Configuração	Modelo	FD001	FD002	FD003	FD004
Base Features	Dummy Regressor	-0,604	-0,251	-2,298	-0,735
	kNN	0,177	0,474	-0,770	0,293
	Linear Regressor	0,409	0,602	-0,900	0,233
	SGD Regressor	0,472	0,632	-0,792	0,197
	Linear SVR	0,409	0,602*	-0,899	0,235*
	DT	0,298	0,589	-0,630	0,346
	RF	0,462	0,695	-0,311	0,424
Polynomial Features	Dummy Regressor	-0,604	-0,251	-2,298	-0,735
	kNN	0,322	0,592	-0,672	0,274
	Linear Regressor	0,409	0,602	-0,900	0,233
	SGD Regressor	0,432	0,612	-0,776	0,262
	Linear SVR	0,413*	0,604*	-0,685*	0,261*
	DT	0,311	0,633	-0,504	0,170
	RF	0,422	0,692	-0,314	0,446

*Teve problemas de convergência
** Não convergiu

Tabela 13. RMSE, usando RUL linear com métodos tradicionais

Configuração	Modelo	FD001	FD002	FD003	FD004
Base Features	Dummy Regressor	52,625	60,162	75,180	71,828
	kNN	37,702	39,010	55,080	45,862
	Linear Regressor	31,950	33,943	57,057	47,758
	SGD Regressor	30,203	32,617	55,419	48,854
	Linear SVR	31,951	33,933*	57,039	47,691*
	DT	34,821	34,462	52,853	44,095
	RF	30,481	29,683	47,403	41,397
Polynomial Features	Dummy Regressor	52,625	60,162	75,180	71,828
	kNN	34,215	34,347	53,529	46,461
	Linear Regressor	31,950	33,943	57,057	47,758
	SGD Regressor	31,319	33,519	55,173	46,840
	Linear SVR	31,831*	33,823*	53,734*	46,887*
	DT	34,484	32,560	50,765	49,677
	RF	31,601	29,844	47,460	40,577

*Teve problemas de convergência
** Não convergiu

C. Valores SHAP na RF

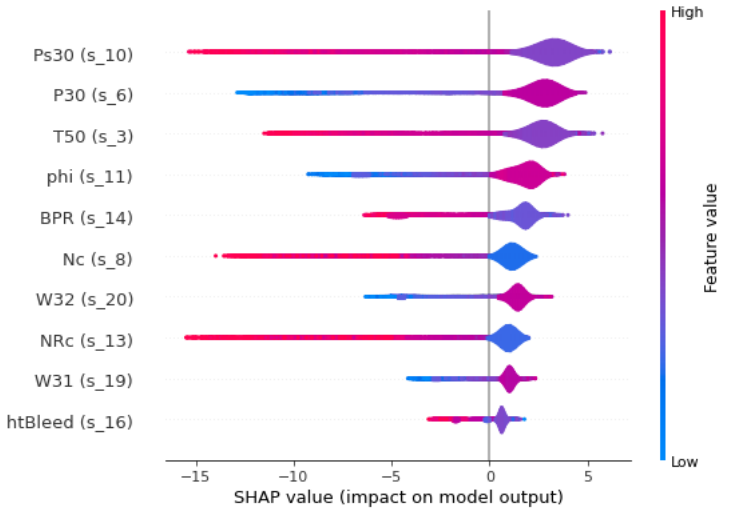

Figura 17. Valores SHAP para o modelo RF no grupo FD001

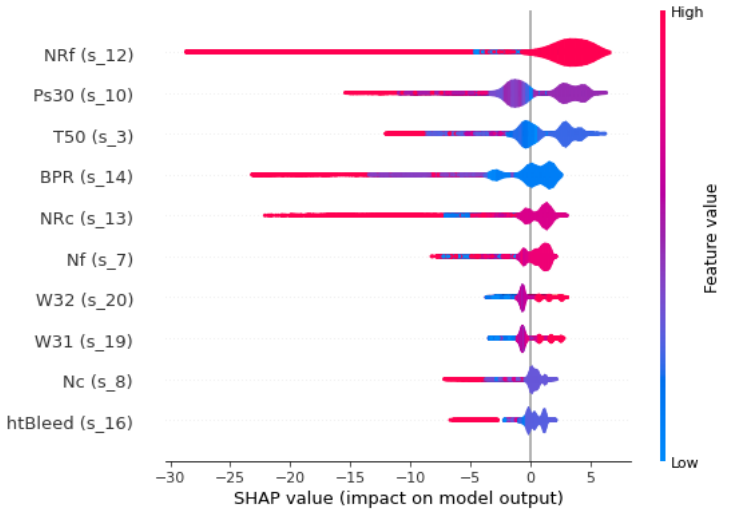

Figura 18. Dez maiores valores SHAP para o modelo RF no grupo FD002

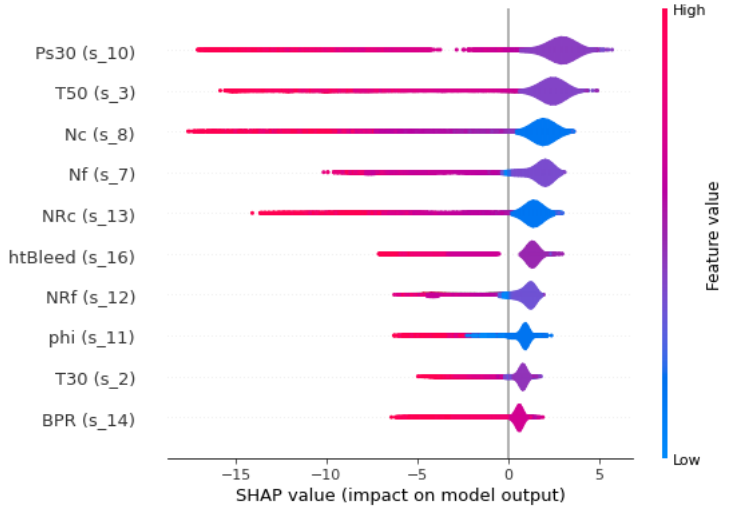

Figura 19. Dez maiores valores SHAP para o modelo RF no grupo FD003

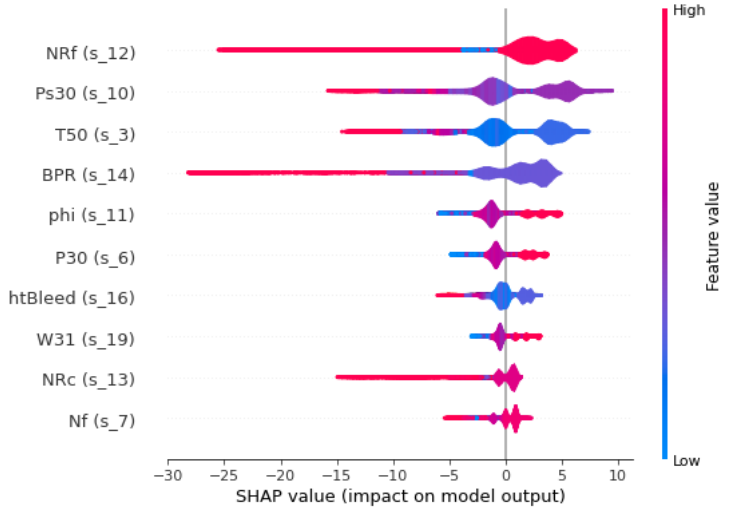

Figura 20. Dez maiores valores SHAP para o modelo RF no grupo FD004