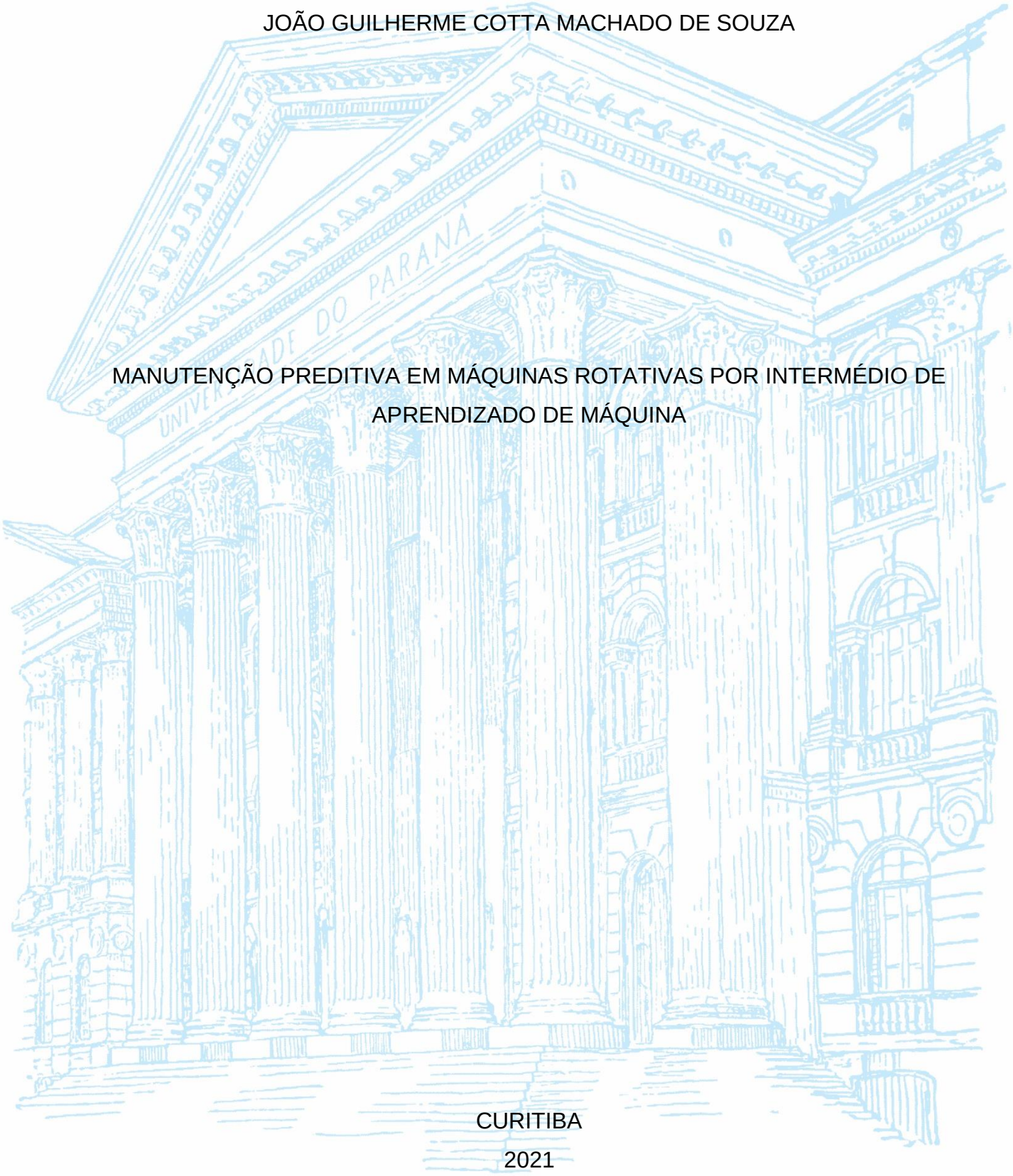
UNIVERSIDADE FEDERAL DO PARANÁ

BRUNO GONÇALVES ROCHA

JOÃO GUILHERME COTTA MACHADO DE SOUZA



MANUTENÇÃO PREDITIVA EM MÁQUINAS ROTATIVAS POR INTERMÉDIO DE APRENDIZADO DE MÁQUINA

CURITIBA

2021

BRUNO GONÇALVES ROCHA
JOÃO GUILHERME COTTA MACHADO DE SOUZA

MANUTENÇÃO PREDITIVA EM MÁQUINAS ROTATIVAS POR INTERMÉDIO DE APRENDIZADO DE MÁQUINA

Trabalho de Conclusão de Curso apresentado ao curso de Graduação em Engenharia Mecânica, Setor de Tecnologia, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Bacharel em Engenharia Mecânica.

Orientadora: Profa. Dra. Giuliana Sardi Venter

CURITIBA
2021

AGRADECIMENTOS

Agradecemos aos nossos familiares, amigos, colegas de turma e professores pelo suporte e incentivo durante a graduação. Um agradecimento especial à nossa professora orientadora, Profa. Dra. Giuliana Sardi Venter pela disponibilidade, suporte e oportunidade prestados a nós.

RESUMO

A aplicação de conceitos e ferramentas da Indústria 4.0 é fundamental para a potencialização da competitividade em empresas. O uso de tecnologias e metodologias como o aprendizado de máquina é fundamental neste processo. Considerado fator significativo na estrutura de custos de uma organização, a manutenção pode e deve ser aprimorada através destes métodos, aplicando-se o que se denomina hoje de manutenção preditiva. Devido a aplicabilidade deste conceito, este trabalho tem como objetivo principal classificar modos de falha em máquinas rotativas por intermédio de aprendizado de máquina e prever possíveis falhas com o algoritmo treinado. Para tal, será utilizado um banco de dados de falhas em máquina rotativa que servirá como entrada para o treinamento da máquina, bem como dados coletados através de experimentos em bancada, que passarão por classificação pelo algoritmo. A fase de experimentação contou com 70 experimentos, realizados com configurações distintas de posicionamento do acelerômetro, rotação e modos de falha. A comparação da precisão obtida por modelos de aprendizado de máquina distintos na classificação do *dataset* permitiu definir o modelo de *Random Forest Classifier* para a classificação final dos experimentos. Para realização desta foi necessária uma generalização do *dataset* disponibilizado, já que este se mostrou altamente específico para o sistema envolvido e resultava em níveis de acurácia insatisfatórios para a classificação de sistemas dissemelhantes. Fundamental para esta etapa foi a análise dos gráficos de aceleração no domínio do tempo e da frequência gerados a partir das informações coletadas em bancada. O modelo selecionado atingiu acurácia de 91,30% na classificação dos experimentos. Atingidos os objetivos definidos, pode-se concluir que é possível classificar modos de falha em máquinas rotativas por intermédio de aprendizado de máquina e aplicar esta ferramenta na manutenção preditiva, prevendo falhas no maquinário e evitando custos adicionais.

Palavras-chave: Manutenção preditiva. Aprendizado de máquina. Máquinas rotativas. *Random Forest Classifier*. Generalização.

ABSTRACT

The application of concepts and tools of Industry 4.0 is fundamental to potentialize the competitiveness in companies. The use of technologies and methodologies such as machine learning is fundamental in this process. Considered a significant factor in an organization's cost structure, maintenance can and should be improved through these methods, applying what is now called predictive maintenance. Due to the applicability of this concept, this work has as main objective classifying failure modes in rotating machines through machine learning and predicting possible failures with the trained algorithm. To achieve this, a machinery fault database will be used, serving as input for the training of the machine, as well as data collected through bench top experiments, which will be sorted by the algorithm. The experimental phase had 70 experiments, carried out with different configurations of accelerometer positioning, rotation and failure modes. The comparison of the accuracy obtained by distinct machine learning models in the dataset classification allowed defining the Random Forest Classifier model for the final classification of the experiments. To perform this, it was necessary to generalize the available dataset, since it was highly specific to the system involved and resulted in unsatisfactory levels of accuracy for the classification of dissimilar systems. Fundamental for this stage was the analysis of acceleration by time and frequency domain plots generated from the information collected on the bench. The selected model reached accuracy of 91.30% in the classification of the experiments. Once the objectives were met, it could be concluded that it is possible to classify failure modes in rotating machines through machine learning and to apply this tool in predictive maintenance, predicting machinery failures and avoiding additional costs.

Keywords: Machine Learning. Predictive maintenance. Rotating machines. Random Forest Classifier. Generalize.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS OU SIGLAS

CBM	Condition Based Maintenance
ISO	International Organization for Standardization
IoT	Internet of Things
RMS	Root Mean Square
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
MAFAULDA	Machinery Fault Database
SVM	Support Vector Machine
LGBM	Light Gradient Boosting Machine
MLP	Multilayer Perceptron

SUMÁRIO

1 INTRODUÇÃO

O cenário global competitivo e de constante transformação se apresenta como desafio e, ao mesmo tempo, oportunidade para empresas dos mais diversos setores. Para estas organizações é imprescindível estar atento ao mercado, às novas tecnologias e às tendências que surgem diariamente e moldam as relações inter- e intra-empresariais. Neste contexto, a Indústria 4.0 é um movimento de transformação que busca a integração da tecnologia com o processo de fabricação e o planejamento estratégico da empresa, com o intuito de aprimorar o processo produtivo e agregar valor ao produto final, com redução de custos e tempo e potencialização de produtividade e competitividade.

Tecnologias, conceitos e metodologias como *Internet of Things*, *Data Analytics*, inteligência artificial e aprendizado de máquina são recorrentes na Indústria 4.0 e representam ferramentas poderosas para o emprego da “produção inteligente” (QIN e CHIANG, 2019).

Importante aspecto dentro deste quadro são as atividades de manutenção de equipamentos, que representam parte significativa dos custos da maioria das empresas e se apresentam, desta forma, como potencial para melhorias através da aplicação dos conceitos da Indústria 4.0. Esta junção deu origem à manutenção preditiva, uma metodologia que apela à prevenção de danos e à previsibilidade de falhas, buscando otimização da eficiência operacional com redução em custos, desvios e atrasos no fluxo de produção (MOBLEY, 2002).

O prospecto de identificar uma falha, antes mesmo que ela ocorra, através da aplicação de aprendizado de máquina, permite a tomada de ações que acarretem no prolongamento da vida útil do equipamento e na redução de custos com troca de peças ou substituição de equipamentos, tornando a manutenção mais assertiva e robusta. A obtenção de um algoritmo capaz de classificar em tempo real falhas em máquinas rotativas se apresenta, neste contexto, como grande diferencial para a indústria.

1.1 JUSTIFICATIVA

Vislumbrando este cenário, é necessário investigar a aplicabilidade de conceitos como Data Analytics e aprendizado de máquina em paralelo com a

manutenção preditiva. Especificamente para este trabalho, é de interesse o estudo da eficácia da classificação de falhas em máquinas rotativas com posterior previsão de ocorrências, aplicando assim o conceito de manutenção preditiva. Além disso, levando em consideração a gama diversa de máquinas rotativas e suas possíveis configurações, é alvo de estudo a precisão com que um modelo classifica um *dataset*, partindo de treinamento e validação realizados para outro *dataset* distinto.

1.2 OBJETIVOS

1.2.1 Objetivo geral

O objetivo principal deste trabalho é classificar modos de falha em máquinas rotativas através do uso de aprendizado de máquina e identificação possíveis falhas ligadas a vibração mecânica usando o algoritmo treinado para reconhecimento.

1.2.2 Objetivos específicos

- Programar algoritmo de aprendizado de máquina.
- Classificar tipos de falhas em máquinas rotativas através de banco de dados com emprego de aprendizagem de máquina.
- Coletar dados reais através de experimentos controlados em bancadas.
- Prever tipos de falhas através de aprendizagem de máquina treinada.
- Ajustar parâmetros de modelos de aprendizado de máquina para otimização da classificação e previsão de falhas.
- Gerar conclusões aplicáveis na indústria para a manutenção preditiva em máquinas rotativas.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 INDÚSTRIA 4.0

Segundo Lasi (2014) o termo Indústria 4.0 engloba uma ampla variedade de conceitos e uma distinção clara e objetiva não é possível em casos individuais. Entre estes conceitos podemos destacar a indústria inteligente, que consiste em uma linha de produção equipada com diversos sensores, atuadores, e sistemas autônomos. Outros conceitos que valem a pena ser citados são: sistemas ciberfísicos (união do físico com o digital), responsabilidade social corporativa, uma produção e distribuição cada vez mais descentralizada, foco voltado para produtos e serviços ofertados de forma individualizada, adaptações para as necessidades humanas (HEINER et al., 2014).

Podemos classificar a Indústria 4.0 como sendo a área responsável pela união de sistemas ciber-físicos conectados com uma combinação de softwares, sensores, processamento e comunicação de dados para gerar informações e agregar valor aos processos de manufatura (BAHRIN et al., 2016). Uma das principais necessidades dessa nova geração industrial é a transformação de máquinas comuns em sistemas de auto-aprimoramento da performance geral e gestão de manutenção (VAIDYA et al., 2018).

De acordo com Lee et.al. (2014) a Indústria 4.0 aponta para a manutenção preditiva como o futuro da indústria, de forma que o maquinário está amplamente conectado com um sistema de monitoramento e prognósticos para tomada de decisões com informações mais precisas (LEE et al., 2014). Dentro deste contexto, uma importante ferramenta da Indústria 4.0, que auxilia na criação de processos de tomada de decisão proativos e não reativos, é a análise de dados, ou *Data Analytics* (QIN e CHIANG, 2019), conceito que será abordado futuramente no transcorrer deste trabalho.

2.2 MANUTENÇÃO PREDITIVA

Os custos com manutenção compõem a maior parte dos custos totais de operação de uma fábrica ou indústrias de bens de produção e, dependendo do setor de operação, esse custo pode variar de 15 a 60% dos custos de bens produzidos.

Pesquisas revelam que um terço dos custos com manutenção são desperdiçados devido a um mau planejamento de manutenção (MOBLEY, 2002). O principal motivo pela ineficiência no processo pode ser relacionado com a falta de dados concretos para quantificar a real necessidade de reparo ou manutenção de um equipamento, a programação de manutenção tem sido, posto que é baseada em dados de monitoramentos de momento ou até mesmo na falha completa da máquina (MOBLEY, 2002).

Segundo Mobley (2002) podemos classificar os principais métodos de manutenção em três tipos: Manutenção '*Run-to-failure'*, Manutenção Preventiva, Manutenção Preditiva. O primeiro e mais antigo método pode ser descrito pela frase "se não está quebrado não arrume", ou seja, esperar pela falha mecânica de um componente ou máquina, para atuar no problema. O grande problema desse método é por ser um reparo não previsto, provocando muitas vezes a paralisação da linha de produção até que a falha seja consertada.

O segundo método é provavelmente o mais popular e aborda a manutenção preventiva, que embora aceite muitas definições diferentes, todas trabalham com o conceito de horas em operação, e a probabilidade que uma máquina tem de falhar após determinado período. A manutenção previamente programada é realizada enquanto o equipamento ainda está funcionando e todos os recursos necessários para esta operação estão disponíveis para serem utilizados (CACHADA et al., 2018).

O terceiro e último método, a manutenção preditiva, será abordado como um dos tópicos principais deste trabalho. É o procedimento mais recente no âmbito de princípios de manutenção e pode ser definido como medidas para prever falhas em um sistema em deterioração, com objetivo de otimizar o processo avaliando o estado de um sistema de forma ampla, com base em dados atuais e históricos do sistema em questão (SELCUK, 2017).

Esse processo pode utilizar de vários conceitos ligados a Indústria 4.0, tais como sensores posicionados estrategicamente para coletar a maior quantidade de dados possíveis, comunicação em tempo real com um banco de dados, ferramentas de análise de dados, aprendizado de máquina, entre outras tecnologias para possibilitar uma decisão inteligente e sistemática para maximizar a interação humano-máquina e minimizar os custos de manutenção (Ana CACHADA et al., 2018).

A manutenção preditiva também pode ser resumida a todas atividades que se utilizam de informações factíveis de um equipamento ou máquina em operação para

aprimorar todo o funcionamento de uma fábrica ou indústria. Para tanto, podem ser utilizadas diversas técnicas e ferramentas de análise para programar todo o plano de manutenção visando sempre o melhor custo benefício para empresa (MOBLEY, 2002).

Essa nova abordagem na manutenção levou a vários estudos e pesquisas para a implementação desses novos conceitos, como por exemplo o CBM - *Condition Based Maintenance*, uma técnica que envolve análise de dados e informações gerados na linha de produção para detectar anormalidades. Esse diagnóstico inclusive apresenta uma norma internacional ISO 13374 que estabelece seis etapas de atuação: aquisição de dados, manipulação dos dados, comparação de dados com indicadores e/ou valores limites, determinação do estado de saúde, previsões futuras, e por fim recomendações de manutenção (CACHADA et al., 2018).

Em outra abordagem Selcuk (2017) ainda divide as análises de manutenção preditiva em 6 tópicos: monitoramento de parâmetros do sistema, análise de vibração, análise de óleo, análise termográfica, análise acústica e demais técnicas. Os principais parâmetros de sistema que devemos ficar atentos no monitoramento de uma máquina são: a eficiência do processo, dissipação do calor, temperatura, corrente elétrica, pressão de fluidos e umidade. É importante ressaltar que apesar de alterações nesses parâmetros serem indicativos do começo de uma falha, também podem ser resultado de um aumento na velocidade ou carga de operação (SELCUK, 2017).

A análise de vibração é uma das principais técnicas utilizadas para monitorar uma máquina rotativa ou cíclica. Através de sensores pode-se monitorar o histórico do nível de vibração ao longo do tempo e identificar defeitos muito antes da falha. Algumas das falhas mais comuns para esse tipo de equipamento são por exemplo desbalanceamentos, desalinhamento vertical e horizontal, defeitos em rolamentos, falhas na rede de energia, eixo torto, folga excessiva, defeitos em pás de turbinas, correias e correntes desajustadas (SELCUK, 2017).

Apesar de muito promissora, os dados obtidos por essa análise não são diretos e precisam passar por diversos tratamentos, processos matemáticos e computacionais, e em alguns casos até mesmo por inteligência artificial para, por fim, poder ser tomada uma decisão assertiva sobre o sistema analisado. Esses métodos são pontos fundamentais que serão abordados com mais detalhe no decorrer deste trabalho.

2.3 DATA ANALYTICS

Davenport (2013) definiu *Data Analytics* como um processo vital que leva a compreensão de conglomerados de informação e que extrai e gera informações valiosas para processos de tomada de decisão e gestão efetiva em organizações. *Data Analytics* busca, de modo geral, entregar as informações certas para a pessoa certa, na hora certa e no formato certo. Atualmente isso está sendo realizado de forma significativamente mais sofisticada (KOOHANG e NORD, 2020). Extrair tendências e padrões significativos e compreender o que os dados “têm a dizer” levaram a uma revolução nas ciências estatísticas (QIN e CHIANG, 2019) e são considerados como aspectos fundamentais para a competitividade, crescimento de produtividade e inovação em empresas no futuro próximo (MANYIKA et al., 2011).

Os conglomerados de informação citados anteriormente, também conhecidos como *Big Data*, são peça elementar da *Data Analytics* e devem, portanto, ser explorados. Qin e Chiang (2019) afirmam que *Big Data* possui quatro fatores distintos: volume, velocidade, variedade e veracidade. O volume, como o nome já diz, está diretamente relacionado à sempre crescente quantidade de informações que as empresas acumulam em seus bancos de dados. A velocidade se faz presente em processos sensíveis ao tempo, em que a agilidade no acesso, interpretação e repasse de informações é necessária. A variedade reflete a capacidade de armazenamento de tipos de dados diferentes, sendo estes dados numéricos, visuais, em forma de texto ou até mesmo em forma de áudio. Por último, mas não menos importante, a veracidade remete à confiabilidade das conclusões obtidas através do conjunto de dados.

Em resumo, *Data Analytics* é o processo de tratamento, interpretação, modelagem e transformação de um conglomerado de dados para que informações valiosas possam ser encontradas e para que o processo de tomada de decisão seja otimizado (VADAPALLI, 2021). Devido ao seu potencial quando aplicada na Indústria 4.0 e, consequentemente na manutenção preditiva, esta ferramenta será utilizada no desenvolvimento deste trabalho. Essa abordagem será o passo inicial para a aplicação do aprendizado de máquina, metodologia complementar base para este trabalho, que será discutida na próxima seção.

2.4 APRENDIZADO DE MÁQUINA

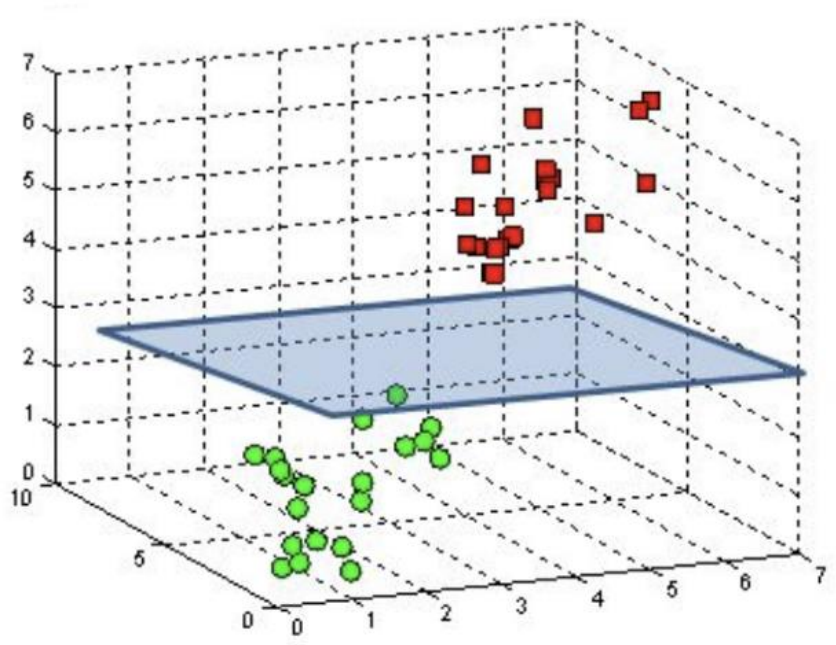
Conforme definido por Singh e Dhiman (2021) aprendizado de máquina é uma metodologia que pode ser aplicada na potencialização de resultados baseado em aprendizado por experiência, que neste caso se refere a dados coletados anteriormente. O treinamento consiste em modificar ou aprimorar o algoritmo baseado inteiramente nas experiências anteriores, sem qualquer influência humana.

Entre outras classificações, o aprendizado de máquina pode ser dividido em supervisionado e não supervisionado. Nos casos em que a variável objetivo é definida em conjunto com os dados de entrada, ou *input data*, tem-se um aprendizado de máquina supervisionado, caso contrário o processo é conhecido como não supervisionado (PATURI e CHERUKU, 2020). Os autores vão além e classificam regressão e classificação como processos supervisionados e *Clustering* e associação como processos não supervisionados. Para este trabalho escolhemos 4 modelos de aprendizagem de máquina para realizar os testes e classificações, sendo eles: *Suport Vector Machine* (SVM), *Random Forest Classifier, Light Gradient Boosting Machine* (LGBM) e *Multilayer Perceptron* (MLP).

2.4.1 *Suport Vector Machine* (SVM)

De forma simplificada esse modelo classifica os dados criando um hiperplano com vetores otimizados para o treinamento que ajuda a classificar os novos dados como mostra a Figura 1 (NAVLANI, 2020). Os hiperplanos podem ser considerados como os limites pelos quais os dados são divididos em diferentes categorias, e a sua dimensão depende da quantidade de características de entrada. Por exemplo, se forem fornecidos ao modelo duas características, o hiperplano será uma linha; já com 3 características será um plano bidimensional e assim por diante (GANDHI, 2018).

FIGURA 1 - Hiperplano bidimensional



FONTE: R. GANDHI, 2018

2.4.2 *Random Forest*

Para explicar o conceito por trás do *Random Forest* é preciso entender como uma *Decision Tree* funciona, já que este modelo é um conjunto de *trees* (árvores). Esta consiste em várias tomadas de decisões binárias (sim ou não) para separar o conjunto de dados em subgrupos assim como na Figura 2, e sua complexidade pode variar conforme a necessidade do problema. Já o *Random Forest* como o próprio nome sugere é um grupo de *Decision Trees,* no qual a predição final será a predição mais comum entre as *trees.* O conceito da eficácia desse modelo se baseia no seguinte princípio: um grande número de modelos (*trees*) relativamente não correlacionados operando como um conjunto terá um desempenho melhor do que qualquer um dos modelos individuais (YIU, 2019).

FIGURA 2 - Exemplo de *Decision tree*

FONTE: (adaptado) YIU, 2019

2.4.3 *Light Gradient Boosting Machine* (LGBM)

Assim como o *Random Forest* o LGBM também é um modelo que utiliza *decision trees* como base, mas possui algumas vantagens e desvantagens em relação ao modelo anterior. Uma destas vantagens é a velocidade computacional, que pode fornecer resultados muito rápidos para grandes *datasets*, porém não funciona muito bem para poucos dados podendo levar a resultados insatisfatórios nesta condição (MANDOT, 2017).

2.4.4 *Multilayer Perceptron* (MLP)

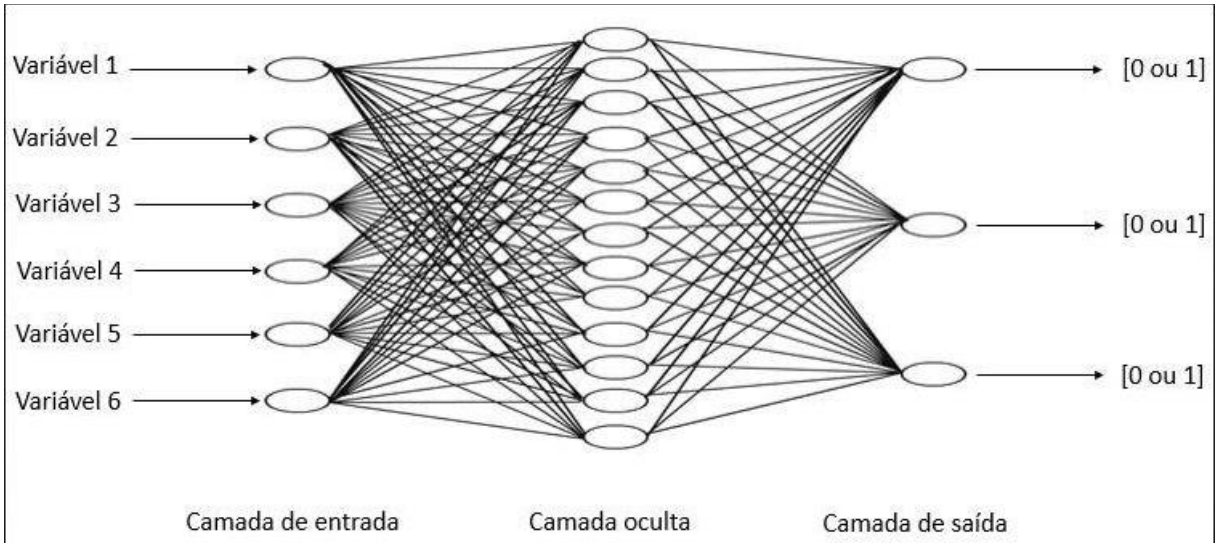
Redes neurais são um subconjunto do aprendizado de máquina que permitem ao algoritmo reconhecer padrões e solucionar problemas utilizando uma estrutura que remete ao comportamento do cérebro humano (KROSE et al., 1996). Também conhecida como rede neural artificial, essa estrutura é composta por camadas de unidades de processamento, contendo uma camada de entrada, uma ou mais camadas escondidas e uma camada de saída. Cada unidade de processamento, ou neurônio artificial, está conectada às outras unidades e recebe valores de peso e limiar.

Quando a camada de entrada é definida e os pesos são atribuídos, cria-se um sinal de saída, que é processado por uma função de ativação, cujo valor será comparado com o limiar da unidade e definirá a ativação ou não do neurônio artificial. Uma vez ativado o sinal é transmitido para a próxima unidade e se torna então um dado de entrada. Este processo iterativo de passagem de informação entre camadas é conhecido como *Feedforward* (BOUGRAIN, 2004).

Retomando o conceito de processos supervisionados definido por Paturi e Cheruku (2020) é possível aprimorar os resultados treinando a rede neural. O treinamento permite que a rede neural ajuste iterativamente os pesos e limiares com o objetivo de minimizar a saída de uma função de perdas. Assim, o processo converge para o mínimo local da função através do método do gradiente descendente (IBM Cloud Education, 2020).

A estrutura definida nos últimos parágrafos corresponde ao perceptron multi-camadas, que será empregado neste trabalho como ferramenta para potencialização da análise de dados para aplicação em manutenção preditiva. A estrutura simplificada desta rede é apresentada pela Figura 3.

FIGURA 3 - Rede neural multicamadas



FONTE: Penedo, 2021

2.5 ANÁLISE DE VIBRAÇÕES EM MÁQUINAS ROTATIVAS

Este trabalho será focado na manutenção preditiva de máquinas rotativas, uma vez que estas representam uma ampla variedade de maquinários presentes na indústria. Além disso, atualmente temos diversos trabalhos e artigos publicados sobre o tema. Mesmo em boas condições, máquinas geram vibrações, muitas relacionadas a eventos periódicos durante sua operação. Através da frequência com que estes eventos se repetem podemos levantar indicativos da fonte dos problemas, relacionados ao eixo de rotação, engrenagens, rolamentos e falhas elétricas por exemplo (RANDALL, 2011).

Há duas razões principais para a vibração ser o ponto de estudo mais abordado nesse tipo de análise. A primeira é que a instrumentação necessária é prática e simples, a segunda é que o diagnóstico apresenta um reflexo compreensivo do estado da máquina (W. LEES, 2016). Além disso, a vibração em uma máquina reage imediatamente a mudanças na operação do maquinário e, portanto, pode ser monitorada tanto de forma permanente quanto intermitente. Isso difere da análise de óleo, por exemplo, que leva vários dias entre a coleta de amostras e sua análise final (RANDALL, 2011).

O ponto fundamental da análise de vibrações em máquinas rotativas é que suas propriedades dinâmicas são funções da velocidade de rotação e muitas vezes funções de outros parâmetros, como pressão ou carga. O sinal gerado por elas pode ser estudado no domínio do tempo ou da frequência (W. LEES, 2016).

2.5.1 Domínio do tempo

Nessa análise os métodos mais comuns e eficientes para caracterizar o nível de vibração de um sistema são: valor de pico (Eq. 1), nível de RMS (*Root Mean Square*) (Eq. 2), o fator de crista, que é definido como a razão entre o valor de pico e o valor de RMS da aceleração (Eq. 3), e a curtose (K), definida pelo quarto momento, normalizado em relação à quarta potência do desvio padrão (Eq. 4) (TANDON, 1999). Os critérios listados acima podem ser definidos pelas equações:

$$valor\ de\ pico = \max |x_i| \tag{1}$$

$$\text{RMS} = \sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}} \tag{2}$$

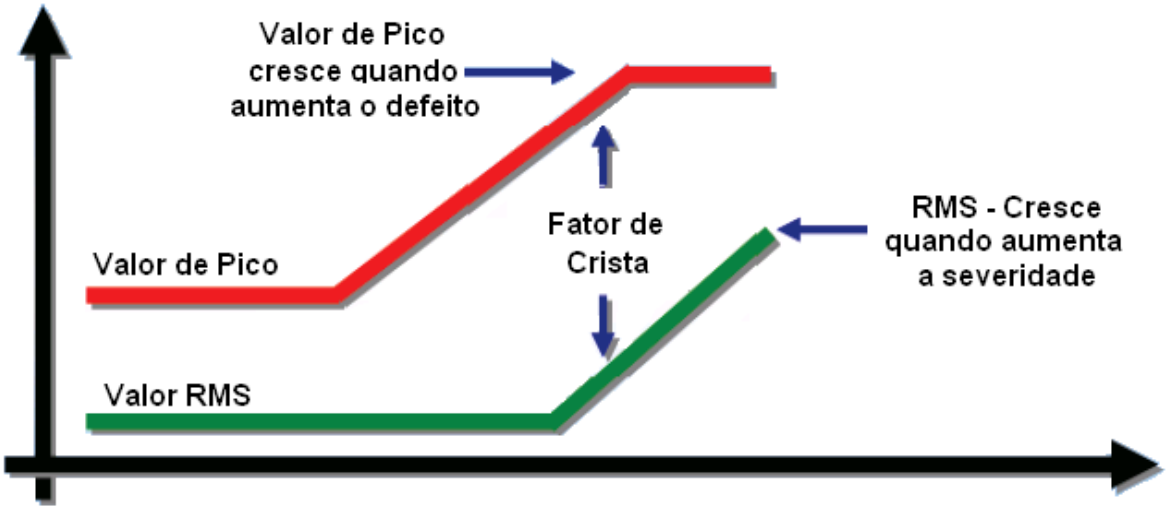
$$Fator\ de\ crista = \frac{valor\ de\ pico}{RMS} \tag{3}$$

$$K = \frac{\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^4}{(\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2)^2} \tag{4}$$

onde $\bar{X} = X_{médio}$

O valor RMS pode ser relacionado ao nível de energia do sinal vibratório e usado para estimar a severidade da vibração. O fator de crista é mais eficiente para detectar falhas iniciais, pois à medida que a falha se propaga, o valor de pico permanece praticamente inalterado, enquanto o RMS cresce, como pode ser observado na Figura 4. Já a curtose pode ser interessante pois a análise não depende da carga e velocidade aplicada ao equipamento (BEZERRA, 2004).

FIGURA 4 – Relação entre Valor RMS, Fator de Crista e valor de Pico



FONTE: MATHIAS, 20—

Outra abordagem simples é comparar os níveis de vibração com padrões para severidade de vibração, como, por exemplo, os estabelecidos pela ISO 10816. Essa norma, com base no tamanho do maquinário, apresenta classificações a partir do RMS de vibração para a velocidade do sinal, como pode ser observado na Figura 5 (RANDALL, 2011).

FIGURA 5 - Padrões de Vibração segundo a ISO 10816

Iso 10816				
			Maquinas grandes	
R.m.s (mm/s)	Classe 1 - Máq. Pequena	Classe 2 - Máq. Média	Classe 3 fundação rigida	Classe 4 - fundação flexivel
0,28	A	A	A	A
0,45				
0,71				
1,12	B			
1,8		B	B	
2,8	C			B
4,5		C	C	
7,1	D			C
11,2		D	D	
18				D
28				
45				
Zona A (Verde) : vibração operacional				
Zona B (Amarelo) : Operação contínua sem restrições				
Zona C (laranja) : Condição é aceitavel apenas por um período limitado de tempo				
Zona D (Vermelho) : vibração perigosos - falha iminente				

FONTE: (adaptado) ISO 10816, 2009

Embora a análise no domínio do tempo tenha suas utilidades, o maior foco desse trabalho irá abordar uma análise espectral, ou seja, no domínio da frequência.

2.5.2 Domínio da frequência

Em essência essa análise requer o uso da transformada discreta de Fourier (DFT), que busca aproximar, numericamente, a transformada analítica de Fourier. Para um conjunto de dados medidos, x_r, igualmente espaçados no tempo, a resposta em frequência, X_k, é definida como:

$$X_k = \sum_{r=0}^{n-1} x_r e^{-j2\pi kr/n} \qquad k = 0,1,2,\dots,n-1 \tag{5}$$

E sua inversa dada por:

$$x_r = \frac{1}{n}\sum_{k}^{n-1} X_k e^{j2\pi kr/n} \qquad r = 0,1,2,\dots,n-1 \tag{6}$$

(W. LEES, 2016).

O cálculo da DFT trabalha com n^2 operações. Porém um algoritmo muito mais rápido, que reduz o número de operações para (n/2)$\log_2 n$, foi desenvolvido sendo chamado de *Fast Fourier Transform* (FFT). Em conjunto com refinamentos subsequentes, ele tornou-se a parte fundamental da análise de vibração e processamento de sinal em geral (W. LEES, 2016).

O indicador básico nessa análise são frequências diretamente ligadas à velocidade de rotação do equipamento, e a presença de amplitude elevada em uma dessas frequências características é um indicador poderoso de provável falha (KIRAL, 2003). Embora uma análise no tempo consiga alertar para uma falha, ainda é evidente que o monitoramento dos espectros de frequência, ao invés dos níveis gerais, terá uma chance melhor de detectar mudanças em qualquer frequência em que ocorram falhas (RANDALL, 2011). Uma análise da FFT de um sinal é uma das principais ferramentas para analisar a amplitude de várias componentes de frequências, uma vez que são conhecidas várias respostas em frequência para algumas falhas mais usuais (MAIS, 2002).

2.5.3 Exemplo domínio do tempo x domínio da frequência

Como ilustração para diferenciar os dois métodos, fizemos uma simulação com auxílio da linguagem de programação *Python* gerando um sinal com as seguintes características:

$$sinal\ gerado = \sum_{i=1} A_i * sen(2\pi f_i t)$$

onde,

$A_i = Amplitude\ do\ sinal\ i\ [\frac{mm}{s^2}]$

$f_i = frequência\ do\ sinal\ i\ [Hz]$

$t = tempo\ em\ segundos\ [s]$

Para este exemplo foram utilizadas as seguintes amplitudes e frequências:

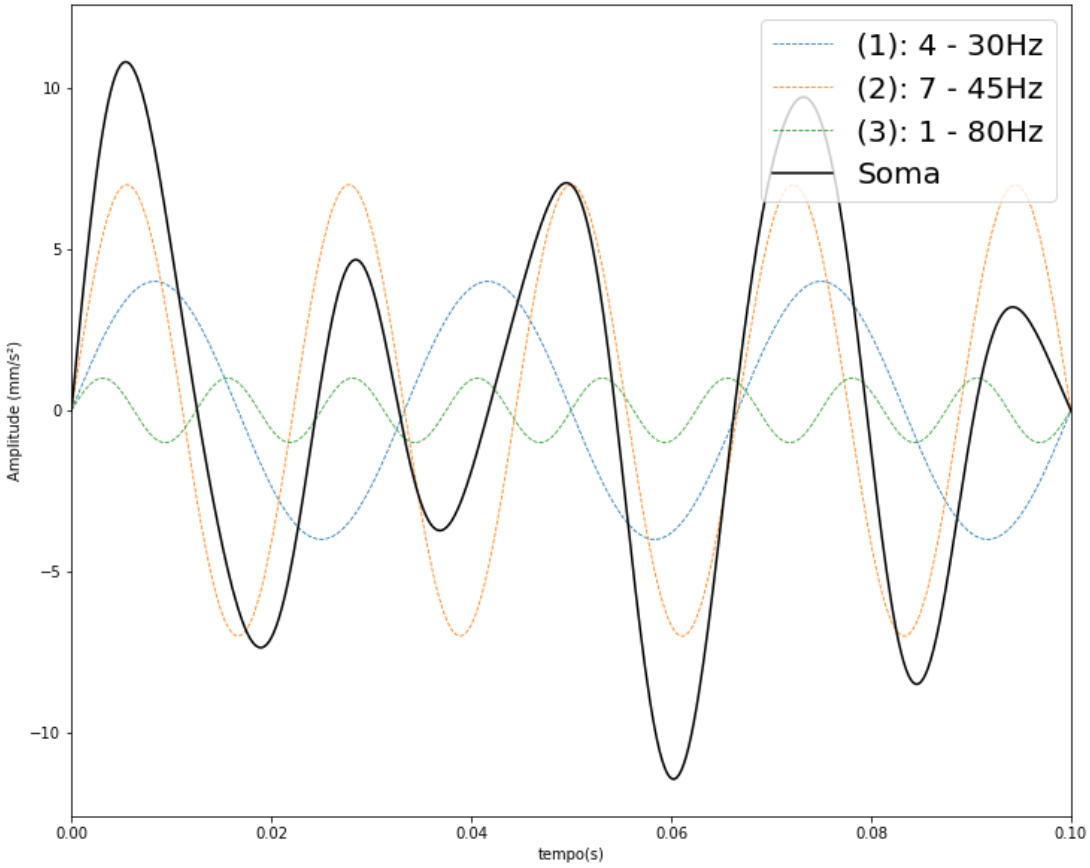
$i = 1 \quad A_1 = 4, \quad f_1 = 30$

$i = 2 \quad A_2 = 7, \quad f_2 = 45$

$i = 3 \;\; A_3 = 1, \;\; f_3 = 80$

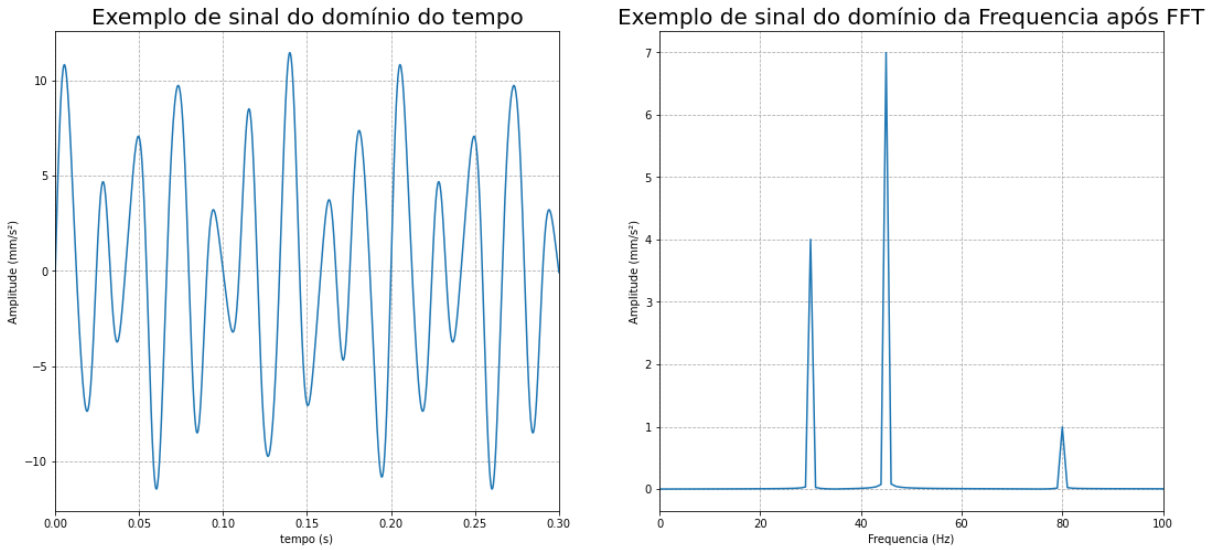
A contribuição de cada sinal pode ser observada na Figura 6. Posteriormente foi realizada a transformação do domínio do tempo para o domínio da frequência através da FFT, e o resultado pode ser constatado na Figura 7.

FIGURA 6 - Contribuição de cada parcela do sinal



FONTE: Os Autores, 2021

FIGURA 7– Domínio do tempo x domínio da frequência.



FONTE: Os Autores, 2021

No gráfico esquerdo da Figura 7, podemos avaliar o estado geral do sinal através de critérios como valor de pico e nível RMS, fator de crista ou curtose, como pode ser observado na Tabela 1. Já em uma análise espectral no domínio da frequência (gráfico direito- Figura 7), é possível ter uma estimativa mais clara das frequências atuando no sinal, sua amplitude e em qual frequência ela é mais relevante, ficando evidentes os picos de 4, 7, 1 mm/s² nas frequências 30, 45 e 80 Hz respectivamente.

TABELA 1 - Resultados do exemplo

Pico	11.438 mm/s²
RMS	5.744 mm/s²
Crista	1.991
Curtose	2.085

FONTE: Os Autores

2.6 FALHAS MAIS COMUNS E COMO MONITORÁ-LAS

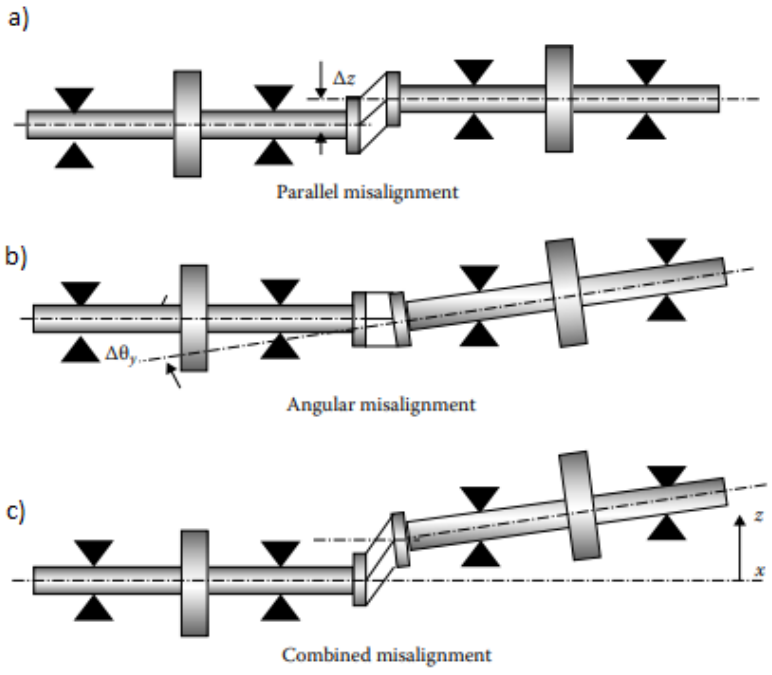
Ao analisar uma máquina rotativa em busca de possíveis falhas, é preciso coletar algumas informações iniciais, como por exemplo a identificação dos componentes envolvidos e suas especificações, a velocidade e condições de

operação, e os tipos de medições que podem ser realizadas no sistema (MAIS, 2002). A seguir serão abordados algumas das principais falhas encontradas em máquinas rotativas e suas respostas espectrais no domínio da frequência, onde será exemplificado como detectá-las a partir de múltiplos da frequência de rotação da máquina, caracterizados por 1x, 2x,3x, ... a frequência de rotação. Por exemplo, uma máquina com velocidade de operação de 1800 rpm (30Hz) é representada por '1x' nessa frequência, 2x descreve a frequência de 2*1800 rpm = 3600 rpm, e assim consequentemente.

2.6.1 Desalinhamento

Acontece quando eixos, acoplamentos e mancais não estão propriamente alinhados, pode ser caracterizado como desalinhamento paralelo, angular ou combinação de ambos. O primeiro ocorre quando os eixos estão paralelos, porém descentralizados (Figura 8.a). O segundo ocorre quando dois eixos são acoplados com uma diferença angular entre os eixos, causando uma força de flexão no eixo (Figura 8.b). Já a combinação dos dois é o caso mais comum e pode ser observada na Figura 8.c (LEES, 2016; MAIS, 2002).

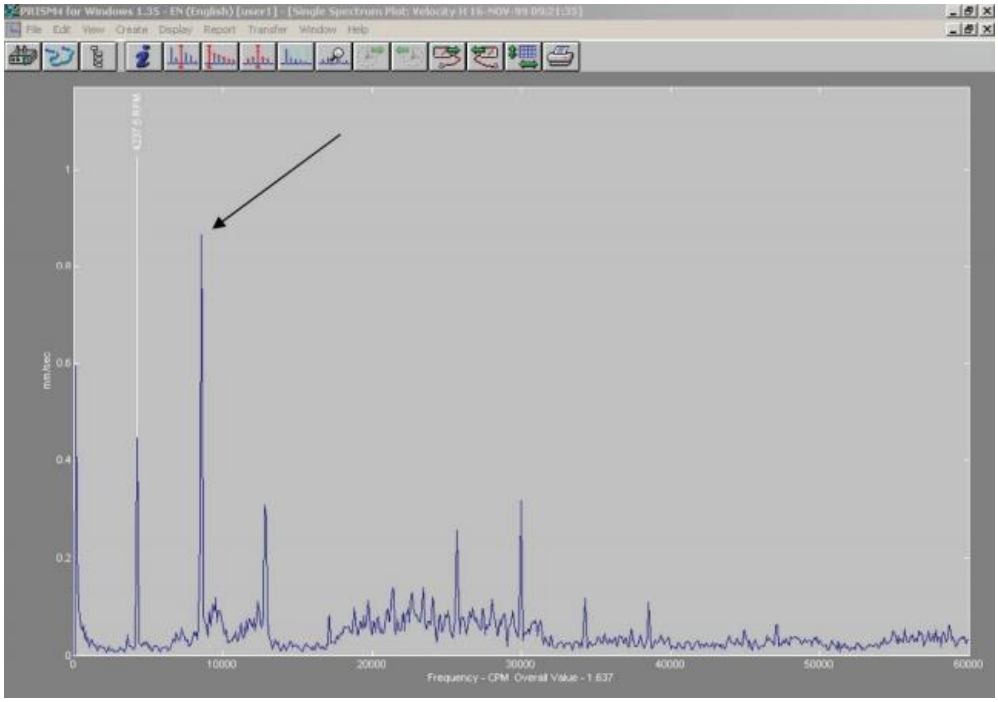
FIGURA 8 - Desalinhamentos: a) Paralelo – b) Angular – c) Misto



FONTE: LEES, 2016

Uma estratégia comum para detectar desalinhamento é verificar amplitudes no espectro FFT das frequências de 1x e 2x a velocidade de rotação. Uma amplitude de até 50% o valor de 1x na frequência de 2x pode ser considerada aceitável, e podendo a máquina operar por um longo período de tempo. Já uma amplitude acima de 50% é um bom indicador de seriedade do desalinhamento e a chance de falha aumenta significativamente, como é o caso observado na Figura 9 (MAIS, 2002).

FIGURA 9 - Espectro FFT demonstra um desalinhamento severo (a amplitude na frequência 2x ~8500 RPM, é quase o dobro da amplitude da frequência de rotação - 4237.5 RPM)



FONTE: MAIS, 2002.

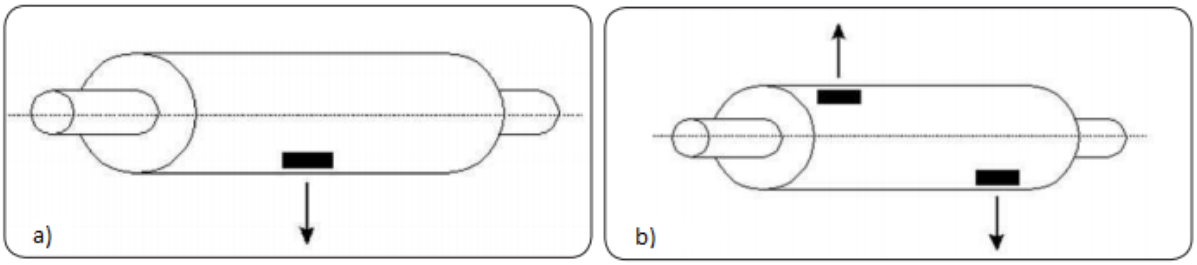
2.6.2 Desbalanceamento de massa

É o defeito mais comum em máquinas rotativas, e se manifesta quando o centro de massa de um sistema ou componente não corresponde com o centro geométrico do mesmo. Ele pode ser dividido em 3 categorias, quais sejam:

- Desbalanceamento estático: ocorre quando temos apenas uma força envolvida no processo, como mostra a Figura 10.a.
- Desbalanceamento casado: ocorre quando temos 2 forças envolvidas em pontos opostos (Figura 10.b), impossibilitando identificar o problema em repouso; porém, com a rotação do motor, fica clara a oscilação do componente.

- Desbalanceamento misto ou dinâmico: Geralmente este é o que ocorre na prática, e é uma combinação dos dois primeiros (LEES, 2016; MAIS, 2002).

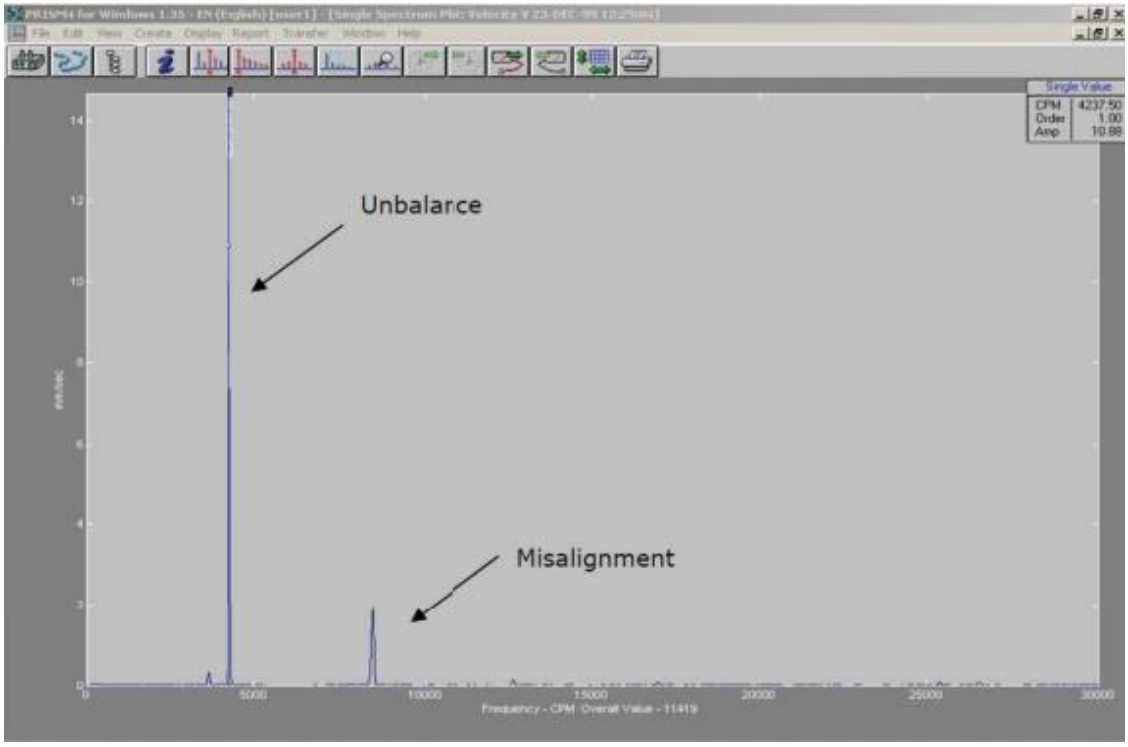
FIGURA 10 - a) Desbalanceamento estático; b) Desbalanceamento casado.



FONTE: MAIS, 2002.

Na análise via FFT (domínio da frequência), podemos identificar o desbalanceamento por uma amplitude mais alta na frequência de rotação do motor (1x) como pode ser observado na Figura 11. Embora não seja o único defeito que gere um aumento na amplitude 1x, os demais defeitos normalmente geram mais picos de amplitude em múltiplos da frequência de operação (1x, 2x, 3x, …) (MAIS, 2002).

FIGURA 11 - Espectro FFT demonstra um desbalanceamento na frequência 1x e um provável desalinhamento na frequência 2x. Frequência de rotação - 4237.5 RPM.

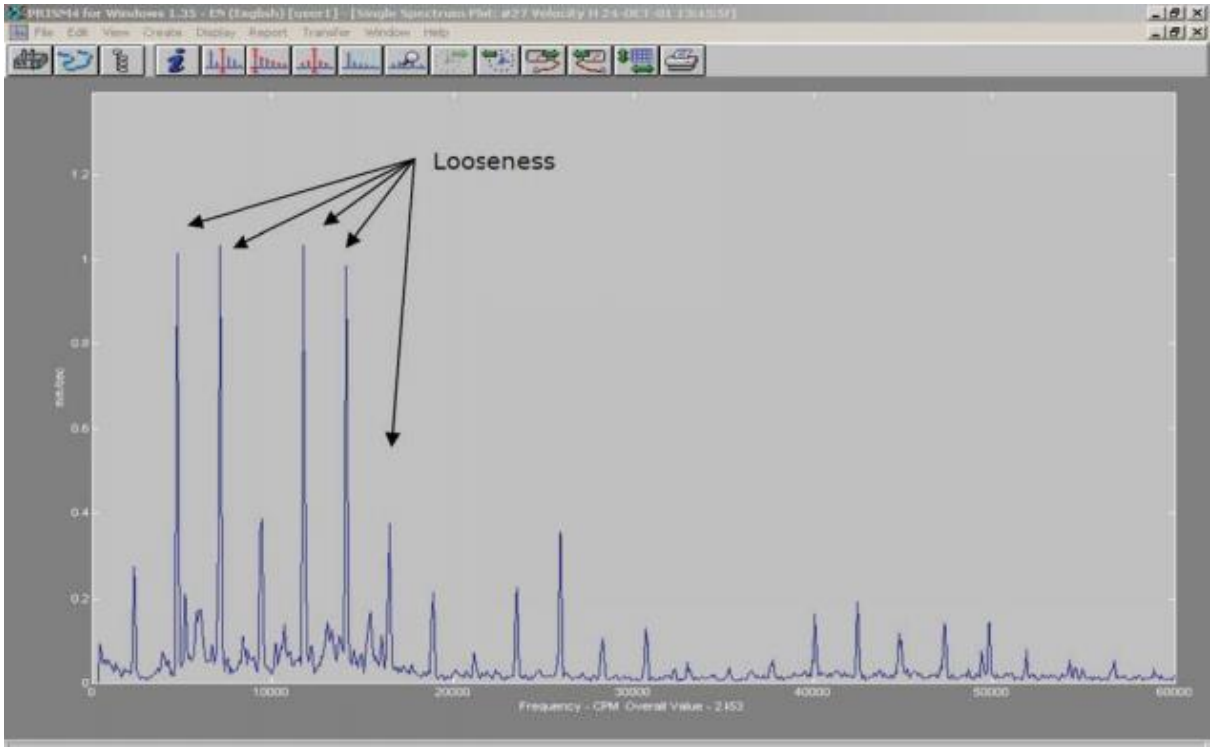


FONTE: MAIS, 2002

2.6.3 Folgas mecânicas

Este defeito ocorre quando componentes não foram projetados adequadamente, ou ainda não foram montados e fixados corretamente, resultando em um encaixe instável e com folgas entre componentes. Em uma análise espectral essa falha pode ser observada por vários picos consecutivos em frequências múltiplas da rotação da máquina, porém podem ser aleatórios e desorganizados, como por exemplo picos de amplitude em 2x, 3x, 4x, 5x, 6x ou ainda 3x, 3.5x, 4x, 5.5x, 6x (MAIS, 2002). Seguindo com a mesma máquina dos exemplos anteriores, podemos observar estes múltiplos em questão na Figura 12.

FIGURA 12 - Espectro FFT demonstra uma provável folga na montagem, com picos em múltiplos da frequência e múltiplos de ½ da frequência de rotação - 4237.5 RPM)



FONTE: MAIS, 2002

2.6.4 Defeitos em rolamentos

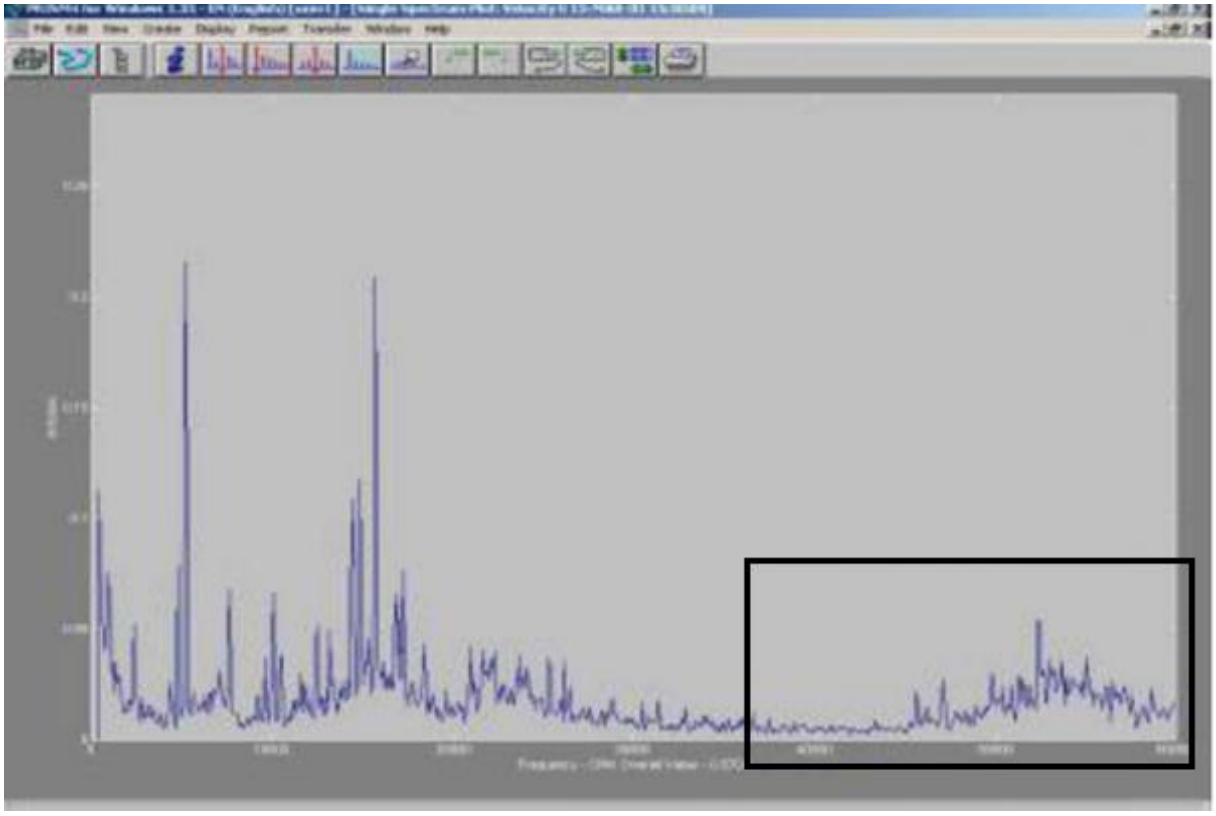
Embora seja um problema muito comum em grandes máquinas, nem sempre eles são a origem do problema, mas sim uma consequência de outros defeitos já

discutidos acima. No entanto, uma lubrificação contaminada ou ineficiente também pode levar a falhas nesses componentes.

O processo de falha ocorre por uma fadiga inicial do rolamento, resultando em tensões de cisalhamento que aparecem ciclicamente imediatamente abaixo da superfície de carga. Com o tempo, essas tensões causam rachaduras que se propagam gradualmente até a superfície. Conforme um elemento rolante passa sobre essas rachaduras, os fragmentos se quebram, aumentando a falha progressivamente e eventualmente torna o rolamento inutilizável.

Em uma análise a partir da FFT, as frequências indicativas de falhas ocorrem em frequências muito mais altas, e com um pico muito menor do que as vistas anteriormente, superiores a 10x a frequência de rotação e de forma consecutiva (MAIS, 2002). A Figura 13 mostra um exemplo de rolamentos defeituosos, porém muito provavelmente gerados por falhas anteriores como desbalanceamento ou desalinhamento.

FIGURA 13 - Espectro FFT demonstra uma provável falha nos rolamentos com picos consecutivos acima de 10x a frequência de rotação - 4237.5 RPM)



FONTE: MAIS, 2002

2.7 SENSORES E AQUISIÇÃO DE DADOS

A vibração de um equipamento pode ser medida de 3 formas: deslocamento, velocidade e aceleração. Para isso são utilizados instrumentos como sensores de proximidade, transdutores de velocidade, acelerômetros ou até mesmo medidores de vibração a laser, que medem deslocamento e/ou velocidade (RANDALL, 2011).

2.7.1 Sensores de proximidade

Sensores de proximidade são sensores que determinam o deslocamento relativo entre o sensor e outra superfície sem entrar em contato com ela. São muito utilizados quando é preciso uma confiabilidade melhor em baixas frequências (abaixo de 10Hz) e geralmente são instalados de forma a medir a vibração em eixos (RANDALL, 2011; GOYAL, 2016).

Eles podem funcionar por indução eletromagnética, onde uma variação da distância altera a resistência do material; por sensores indutivos, nos quais objetos metálicos ao se aproximarem da extremidade de uma bobina alteram a indutância; ou sensores capacitivos que medem a variação da capacitância conforme a distância entre o sensor e o objeto muda (GOYAL, 2016).

2.7.2 Transdutores de velocidade

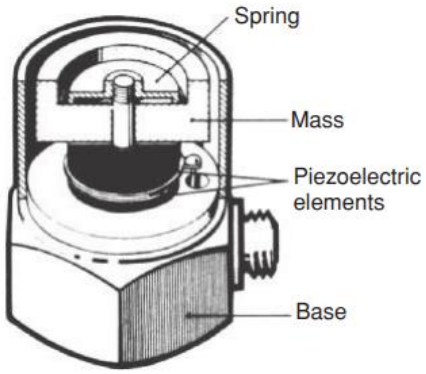
São transdutores que fornecem um sinal proporcional à velocidade absoluta. São efetivamente bobinas de alto-falantes invertidas, e operam segundo o princípio de indução eletromagnética. São geralmente utilizados para frequências entre 10-1000 Hz, balanceamento e monitoramento em máquinas rotativas (RANDALL, 2011; GOYAL, 2016).

2.7.3 Acelerômetros

São os instrumentos de medição de vibração mais utilizados e consistem em transdutores piezoelétricos que produzem um sinal proporcional à aceleração absoluta. Os elementos piezoelétricos geram uma carga elétrica proporcional à

deformação gerada, por exemplo, via sistema massa-mola, como pode ser observado na Figura 14.

FIGURA 14 - Esquemático de um acelerômetro padrão



FONTE: RANDALL, 2011

Quando o acelerômetro for conectado a uma superfície de um objeto vibrando, a massa é forçada a se movimentar, gerando deformações nos elementos piezoelétricos proporcionais à variação da aceleração (RANDALL, 2011). As principais vantagens dos acelerômetros são por serem equipamentos compactos e leves, podendo medir com precisão uma ampla faixa de frequências (GOYAL, 2016).

2.7.4 Medidores de vibração a laser

Em tempos mais recentes, foram desenvolvidos transdutores baseados no princípio do laser de Doppler, no qual um feixe de laser é refletido de uma superfície vibrando e sua frequência é determinada de acordo com a velocidade absoluta da superfície pelo efeito Doppler. Uma das principais vantagens desse método é que o equipamento não precisa entrar em contato com a superfície do objeto analisado, e o ponto de medição pode ser alterado rapidamente. Porém o seu alto custo de investimento limita sua utilização em larga escala (RANDALL, 2011).

2.8 PROCESSAMENTO DO SINAL

Segundo Bjørnø (2017), processamento de sinais condensa medições para extrair informações a respeito de algum estado da natureza. O autor afirma ainda que embora as medidas sejam adquiridas ao mesmo tempo no conjunto espaço-tempo, o

processamento de sinais geralmente é dividido para combinar primeiramente os sinais advindos do espaço e, então, aqueles oriundos do tempo. Simplificando, processamento de sinais é um campo de estudo que foca na análise e transformação de sinais, cujos processos tem o objetivo de otimizar a transmissão, qualidade e armazenamento das medições.

Conforme definido nos objetivos, o foco deste trabalho está na aplicação de aprendizado de máquina para classificação e previsão de falhas em máquinas rotativas e, embora o processamento de sinais faça parte das etapas que envolvem experimentos, a teoria envolvida não será aprofundada. De toda forma como discutido na Seção 2.5.2 deste estudo, ressalta-se que a FFT é essencial para a disciplina de processamento de sinais, permitindo a análise de sinais em função da frequência.

3 MATERIAL E MÉTODOS

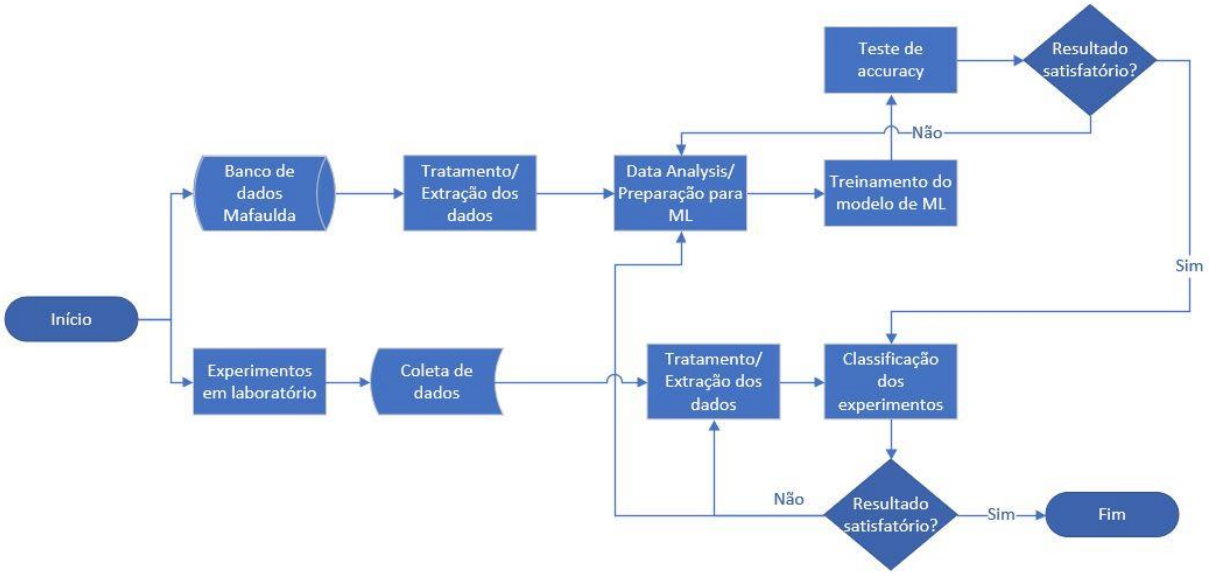
3.1 ROTEIRO GERAL

Tendo em vista o objetivo geral deste trabalho, foi definida uma metodologia que contempla 3 etapas principais. A etapa inicial envolve o estudo de um banco de dados de falhas em máquinas, do inglês *Machinery Fault Database* (MAFAULDA), com posterior tratamento de dados e adequação ao emprego do aprendizado de máquina, que se configura como segunda etapa. Nesta fase será desenvolvido um código em linguagem *Python*, cujo objetivo é a classificação de tipos de falhas em máquinas rotativas, que utilizará o banco de dados citado anteriormente como entrada para treinos e testes. Com o intuito de comprovar a eficácia do algoritmo será realizada uma etapa final de testes em laboratório, buscando replicar as condições aplicadas nos experimentos realizados pela equipe que disponibilizou o MAFAULDA, garantindo assim a obtenção de um novo *dataset*. Este servirá como entrada para a máquina, que mais uma vez classificará os tipos de falha envolvidos.

As seções a seguir apresentarão em mais detalhes as 3 etapas e trarão informações importantes para o desenvolvimento e obtenção dos resultados práticos deste trabalho. Neste estão inclusos os equipamentos da bancada, sistemas de medição e monitoramento, condições dos experimentos e o setup montado para a primeira e terceira etapas, bem como as bibliotecas, ferramentas de *Data Analysis* e aprendizado de máquina, *hypertuning* e as métricas de avaliação de desempenho para a segunda etapa.

O panorama geral do desenvolvimento deste trabalho está apresentado na Figura 15 a seguir.

FIGURA 15 - Fluxograma de atividades



FONTE: Os Autores, 2021

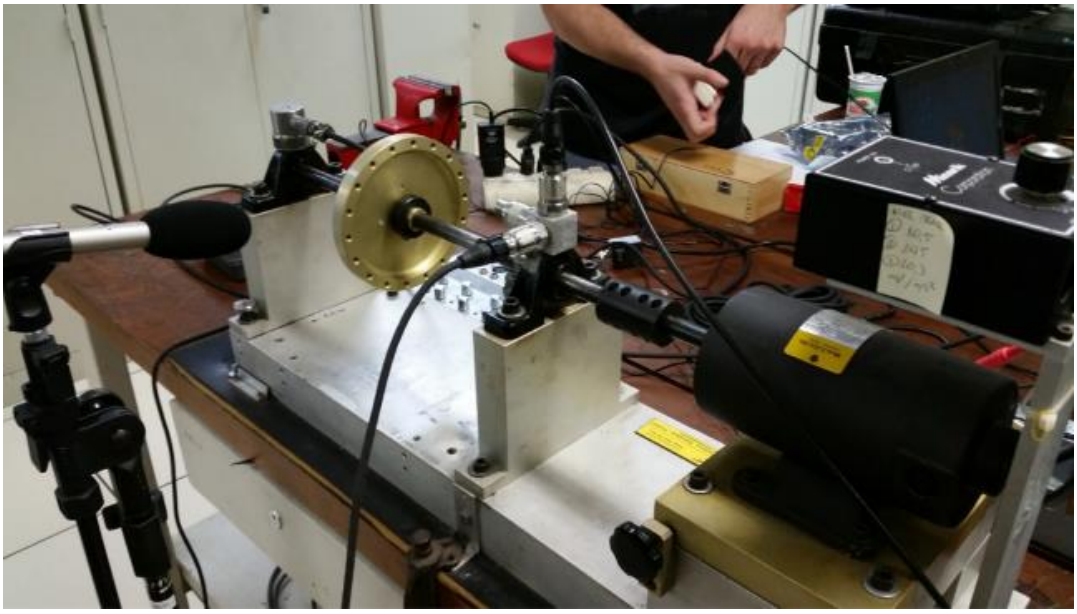
3.2 MAFAULDA

Para treinar e otimizar o algoritmo de aprendizado de máquina utilizamos uma base de dados pública chamada MAFAULDA. Este conjunto de dados foi disponibilizado pela Coppe – Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia, da Universidade Federal do Rio de Janeiro, e consiste nos resultados de monitoramento de uma máquina rotativa no domínio do tempo em diferentes condições de operação. No total foram disponibilizados 1951 arquivos agrupados em 6 possíveis estados: normal, desbalanceado, desalinhamento horizontal, desalinhamento vertical, falhas de rolamento interna e externa.

3.2.1 Equipamentos da bancada de ensaio

Para a obtenção desses dados a Coppe UFRJ desenvolveu uma bancada de ensaio (Figura 16). Apesar de não conseguirmos acesso a todos os detalhes desse projeto foram disponibilizados algumas informações dos componentes principais, como mostra a Tabela 2.

FIGURA 16 - Bancada de Experimentos MAFAULDA



FONTE: M.A. Marins et al./ Journal of the Franklin Institute 355, 2018

TABELA 2 - Equipamentos bancada de ensaio

Equipamento		**Valor**	**Unidade**	**Quantidade**
Motor		1/4	CV	1
Eixo	diâmetro	16	mm	1
	comprimento	520	mm	
Mancais	nº de esferas	8		2
	diâmetro esferas	0,7145	cm	
	diâmetro gaiola	2,8519	cm	

FONTE (adaptado): MAFAULDA, disponível em:
<http://www02.smt.ufrj.br/~offshore/mfs/page_01.html>

3.2.2 Equipamentos de medição/monitoramento

Já para o monitoramento e coleta de dados foram utilizados acelerômetros, um tacômetro, um microfone e um modulo de aquisição de sinal conforme a Tabela 3.

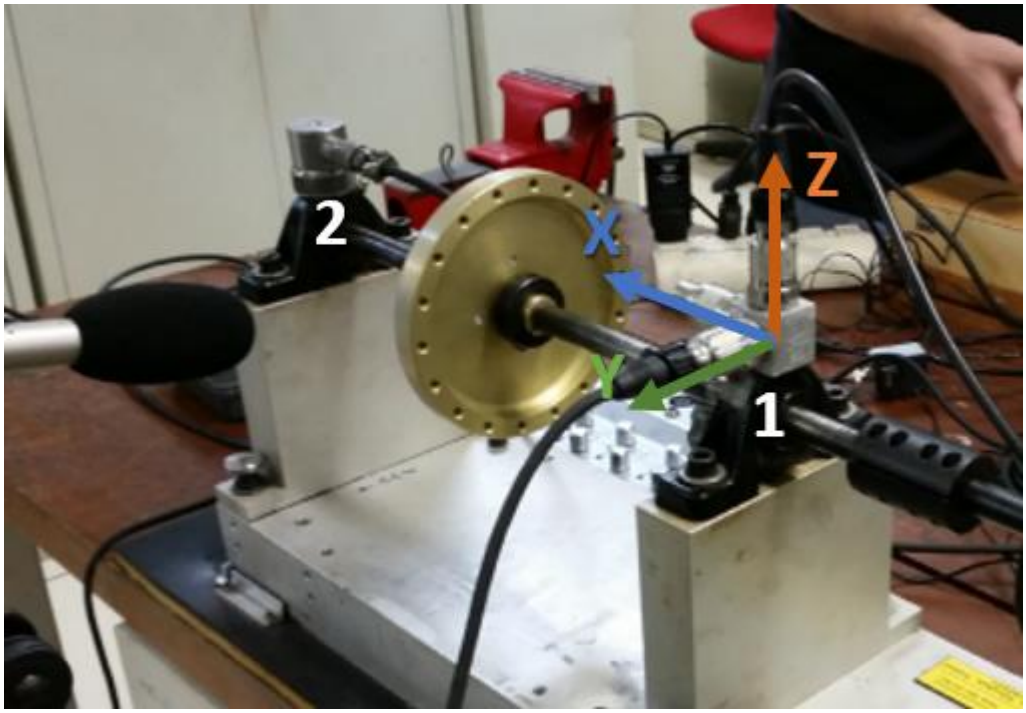
TABELA 3 - Equipamentos de monitoramento

Equipamento	**Modelo**	**Características**		**Quantidade**
Sensor Industrial IMI	601A01	sensibilidade	(±20%) 10.2 mV por m/s2	3
		Faixa de frequência	(±3 dB) 0.27-10.000 Hz	
		Faixa de medida	±50 g (±490 m/s2)	
Sensor Industrial IMI triaxial	604B31	sensibilidade	(±20%) 10.2 mV por m/s2	1
		Faixa de frequência	(±3 dB) 0.5-5.000 Hz	
		Faixa de medida	±50 g (±490 m/s2)	
Tacômetro analógico	MT-190			1
Microfone	Shure SM81			1
Modulo de aquisição de sinal	NI 9234	nº de canais	4	2
		frequência de amostragem	51,2kHz	

FONTE (adaptado): MAFAULDA, disponível em:
<http://www02.smt.ufrj.br/~offshore/mfs/page_01.html>

Para facilitar a compreensão iremos nos referir aos acelerômetros como acelerômetro 1 ou 2, sendo o primeiro o mais próximo do motor. Já suas direções de medição serão referidas como X, Y e Z correspondendo as direções axial, transversal e radial respectivamente em relação ao mancal que está posicionado o acelerômetro, como mostra a Figura 17.

FIGURA 17 - Direções XYZ MAFAULDA



FONTE (adaptado): M.A. Marins et al, 2018

3.2.3 Configurações de operação

Além da condição normal de operação, na qual a bancada estava totalmente alinhada e balanceada, foram simuladas falhas comuns em máquinas rotativas, como desbalanceamento, desalinhamento vertical e horizontal, e defeitos em rolamentos. Para cada uma dessas configurações ainda foi alterada a velocidade de operação do motor variando de 737 rpm até 3686 rpm com passos aproximados de 60 rpm.

Para a condição desbalanceada foram adicionados pesos em um disco conectado ao eixo. Os pesos inseridos foram de 6g, 10g, 15g, 20g, 25g, 30g e 35g. Já na configuração de desalinhamento o motor foi deslocado tanto na horizontal quanto na vertical através da adição de calços ou ajustes com parafusos nas laterais, variando o deslocamento de 0.5mm a 2mm fora da posição central.

Para os defeitos nos rolamentos foram realizados ensaios com apenas um mancal defeituoso por vez e as falhas simuladas foram: defeitos externos, internos ou nos rolamentos. Segundo os autores do experimento estas falhas são praticamente imperceptíveis quando não há nenhum desbalanceamento, em razão disso eles adicionaram massas de 6g, 20g e 35g de forma a induzir uma falha detectável.

O resumo e a quantidade de medições de cada configuração de operação podem ser observados na Tabela 4.

TABELA 4 - Configurações de operação

Operação	**Medidas**
Normal	49
Desalinhamento Horizontal	197
Desalinhamento Vertical	301
Desbalanceamento	333
Falhas no rolamento 1	
Internas	188
Externas	184
Esferas rolantes	186
Falhas no rolamento 2	
Internas	188
Externas	188
Esferas rolantes	137
TOTAL	1951

FONTE (adaptado): MAFAULDA, disponível em:
<http://www02.smt.ufrj.br/~offshore/mfs/page_01.html>

3.2.4 Dados obtidos

Os dados foram coletados a partir de um módulo de aquisição de sinal especificado na Tabela 3, sendo amostrado com uma frequência de 50kHz durante 5s resultando em uma amostra de 250.000 pontos para cada sinal. Para cada uma das 1951 configurações descritas acima foi gerado um arquivo com 8 colunas, sendo elas: sinal de um tacômetro analógico para estimar a frequência de rotação do motor, sinais de aceleração no domínio do tempo nas direções axial, radial e tangencial para os 2 mancais e por fim uma coluna com o resultado do sinal de um microfone, essa última não será considerada no decorrer desse trabalho.

3.3 ANÁLISE DOS DADOS E APRENDIZADO DE MÁQUINA

Após uma análise inicial do MAFAULDA ficou claro que não seria viável trabalhar com todos os dados apresentados, já que estes somavam mais de 13 GB ao total e o treinamento de aprendizado de máquina seria lento e difícil de otimizar. Desse modo foi preciso fazer uma seleção das melhores *features* (propriedades) de cada arquivo. Como base para esta etapa utilizamos tanto ferramentas populares de aprendizado de máquina quanto a teoria abordada na fundamentação teórica.

3.3.1 Extração de *features*

Para esta etapa decidimos extrair as propriedades mais importantes no domínio do tempo e da frequência como foi apresentado na seção 2.5 ANÁLISE DE VIBRAÇÕES EM MÁQUINAS ROTATIVAS. Portanto para cada configuração de operação, através do Apêndice 2, foram aferidas as seguintes características:

- Rotação do motor (obtida a partir do sinal do tacômetro analógico)
- Propriedades no domínio do tempo
    - Pico (Eq. 1)
    - RMS (Eq. 2)
    - Fator de crista (Eq. 3)
    - Curtose (Eq. 4)
- Propriedades no domínio da frequência:
    - Amplitude na frequência 1x (1 * rotação do motor)
    - Amplitude na frequência 2x
    - Amplitude na frequência 3x
    - Amplitude na frequência 4x
    - Amplitude na frequência 5x
    - Amplitude na frequência 6x

Como temos 6 colunas com sinais diferentes de aceleração, as *features* acima, com exceção da rotação, foram extraídas para cada um dos sinais, totalizando 61 *features* e um *status* (condição da máquina que será classificada por aprendizado de máquina) como está representada na Figura 18.

FIGURA 18 - Tabela de *features* e *status*

FONTE: Os Autores, 2021

Para o status da máquina temos 10 condições possíveis, representadas na Tabela 5.

TABELA 5 - Possíveis estados de operação da Máquina

Status	
	Falha no rolamento
Normal	NÃO
Normal	internas
Normal	externas
Normal	elementos rolantes
Desbalanceamento	NÃO
Desbalanceamento	internas
Desbalanceamento	externas
Desbalanceamento	elementos rolantes
Desalinhamento Vertical	NÃO
Desalinhamento Horizontal	NÃO

FONTE: Os Autores, 2021

Como já foi dito anteriormente, este banco de dados conta com 1951 arquivos, cada um equivalente a uma medida de 5 segundos. Porém para aumentar a quantidade de dados e sabendo que 1 segundo de amostragem era o suficiente para representar cada medida, optamos por dividir cada arquivo em 5, um para cada segundo de medição, terminando, assim, com 9755 dados.

3.3.2 Ferramentas de DA e preparação para aprendizado de máquina

Assim como na maior parte prática deste trabalho utilizamos a linguagem de programação em *Python* para auxiliar nessa tarefa, mais especificamente a biblioteca Pandas, que oferece diversas funcionalidades e ferramentas para análise de dados, dentre as quais podemos destacar:

- Leitura e manipulação de dados com muita eficiência;
- Análises estatísticas
    - Detecção de *outliers* (pontos fora do padrão);
    - Matriz de Correlação de Pearson (indica o grau de correlação entre variáveis);
- Separação dos dados em *features* e *status* (coluna do banco de dados que estamos querendo prever com aprendizado de máquina).

Outra ferramenta que utilizamos foram os filtros digitais, que permitiram filtrar frequências indesejadas dos resultados adquiridos. Para isso utilizamos os filtros *butterworth* disponibilizados pelo *scipy*.

3.3.3 Aprendizado de máquina

Nesse tópico serão abordados modelos, técnicas e ferramentas para a otimização do treinamento de aprendizado de máquina. Utilizaremos 4 modelos de aprendizagem de máquina para classificar os dados obtidos, sendo eles: *Suport Vector Machine* (SVM), *Ligth Gradient Boosting Machine Classifier* (LGBM Classifier), *Random Forest Classifier* e *Multilayer Perceptron (*MLP)*.*

3.3.3.1 *StandardScaler*

Visto que trabalhamos com múltiplas variáveis e que cada uma destas variáveis apresenta faixa de valores distinta, é necessário aplicar uma padronização dos dados com emprego da ferramenta *StandardScaler*. Este recurso remove a média e escala para a variância, ambas as operações para as amostras de treino, de acordo com a Equação 7 a seguir.

$$z = \frac{(x - u)}{s} \tag{7}$$

em que x é a amostra a ser normalizada, u é a média das amostras de treino e s a sua variância. Cada uma das variáveis passa por esse processo de centralização e escala, que é considerado requisito fundamental para algoritmos de aprendizado de máquina, tendo em vista que estes podem se comportar de forma anômala caso as variáveis não se aproximem de uma distribuição normal de dados.

3.3.3.2 Função de ativação

A função de ativação realiza a transformação não-linear do sinal de entrada dos neurônios, o que permite que a rede neural atualize os pesos e *bias* de forma não linear e aprenda com a informação disponibilizada. Ela é responsável pela decisão de ativação de um neurônio, ou seja, se a informação de entrada é relevante ou não. Uma rede neural sem função de ativação realiza apenas regressão linear. Entre as funções mais recomendadas para tarefas de classificação, caso deste estudo, a que fornece os melhores resultados foi encontrada através de *Hyperparameter Tuning*, que será detalhado em uma seção posterior.

3.3.3.3 *Dropout*

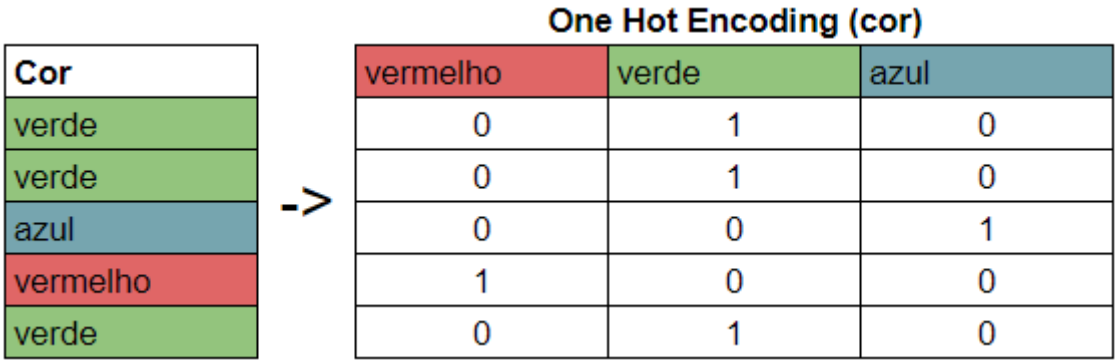
Para compreender a técnica de *Dropout* é necessário primeiramente compreender o conceito de *Overfitting. Overfitting* ocorre quando o modelo treina por tempo prolongado e inclui ao aprendizado ruídos ou informações relevantes dentro do conjunto de dados. Quando isso ocorre a rede neural não é capaz de generalizar com informações novas e, desta forma, não executa tarefas de classificação e predição com precisão.

A técnica de *Dropout* reduz o efeito do *Overfitting o*mitindo de forma aleatória neurônios das camadas da rede neural, fazendo com que sua contribuição para a ativação de neurônios subsequentes seja nula e atualizações de pesos e bias não sejam aplicadas. O parâmetro considerado neste caso é *rate*, que representa a frequência com a qual neurônios serão desconsiderados durante o treinamento.

3.3.3.4 *One Hot Encoding*

O algoritmo de MLP empregado neste trabalho requer que os dados tenham valores numéricos, embora a classificação final a ser realizada se baseie em dados categóricos como “normal”, “horizontal”, “vertical” e “desbalanceamento”. Desta forma, é necessário transformar estas categorias em inteiros que sejam interpretáveis pela rede neural. Para tal utiliza-se o método de *One Hot Encoding,* que converte os dados categóricos em um vetor binário de 0s e 1s. A Figura 19 abaixo exemplifica a aplicação do *One Hot Encoding* para o caso de cores.

FIGURA 19 - Exemplo de *One Hot Encoding*



FONTE: Os Autores, 2021

3.3.3.5 *Early Stopping*

Esta ferramenta monitora a precisão da etapa de validação e interrompe o treinamento caso este valor não aumente em certo valor por um número específico de épocas, ambos pré-determinados pelo usuário. A aplicação deste método evita o *overfitting*, caso em que muitas épocas são determinadas para o treinamento, e simultaneamente a obtenção de um modelo de aprendizagem *underfitting*, em que poucas épocas são empregadas.

3.3.4 Métricas de avaliação

Buscando qualificar os resultados obtidos pela classificação, é necessário definir métricas de avaliação dos modelos de aprendizado de máquina. Comumente são levadas em consideração a precisão do modelo e a complexidade computacional, porém este trabalho focou na primeira métrica, chamada de *accuracy,* que representa, simplificadamente, a frequência com que o algoritmo classifica corretamente o tipo de falha observado na máquina rotativa. A velocidade de classificação é importante para o caso de monitoramento online, porém a classificação pós treinamento ocorre de forma rápida, motivo pelo qual não será dado tanto peso à complexidade computacional.

3.3.5 Modelos de validação dos dados

O processo de classificação executado por um algoritmo envolve a etapa de treinamento e uma etapa subsequente de testes. É nessa etapa que é avaliada a performance de treinamento do algoritmo, sendo esta divisão do conjunto de dados denominada modelo de validação dos dados. Neste trabalho utilizamos dois modelos distintos, que serão abordados nas próximas seções.

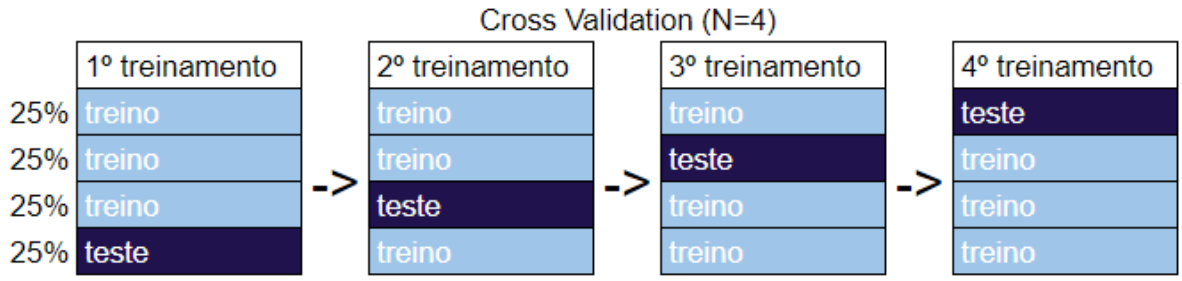
3.3.5.1 *Train-Test Split*

Para evitar o chamado *overfiting* dos dados uma técnica muito utilizada é o *train-test-split*, no qual uma porcentagem dos dados originais é separada do treinamento sendo chamada assim de conjunto de teste ou validação. Esse grupo de dados é utilizado somente após o algoritmo de aprendizado de máquina concluir o treinamento com o restante dos dados, para em sequência serem avaliados como um conjunto de dados novos. Uma separação usual dos dados de teste é de 20% a 30% do total dos dados.

3.3.5.2 *Cross Validation*

Outra técnica para avaliar com uma maior confiabilidade a performance de um modelo de aprendizado de máquina é a validação cruzada. Ela divide o conjunto de dados em N partes e faz N treinamentos, onde N é definido pelo usuário. Para cada treinamento uma das partes é utilizada como teste e todas as outras são utilizadas para treinamento. No final do processo essa técnica terá utilizado todos os dados como treino e teste, como mostra a Figura 20 para um N=4. O resultado dessa técnica é a média de todos os treinamentos para a métrica de avaliação escolhida.

FIGURA 20 - Exemplo *Cross Validation*



FONTE: Os Autores, 2021

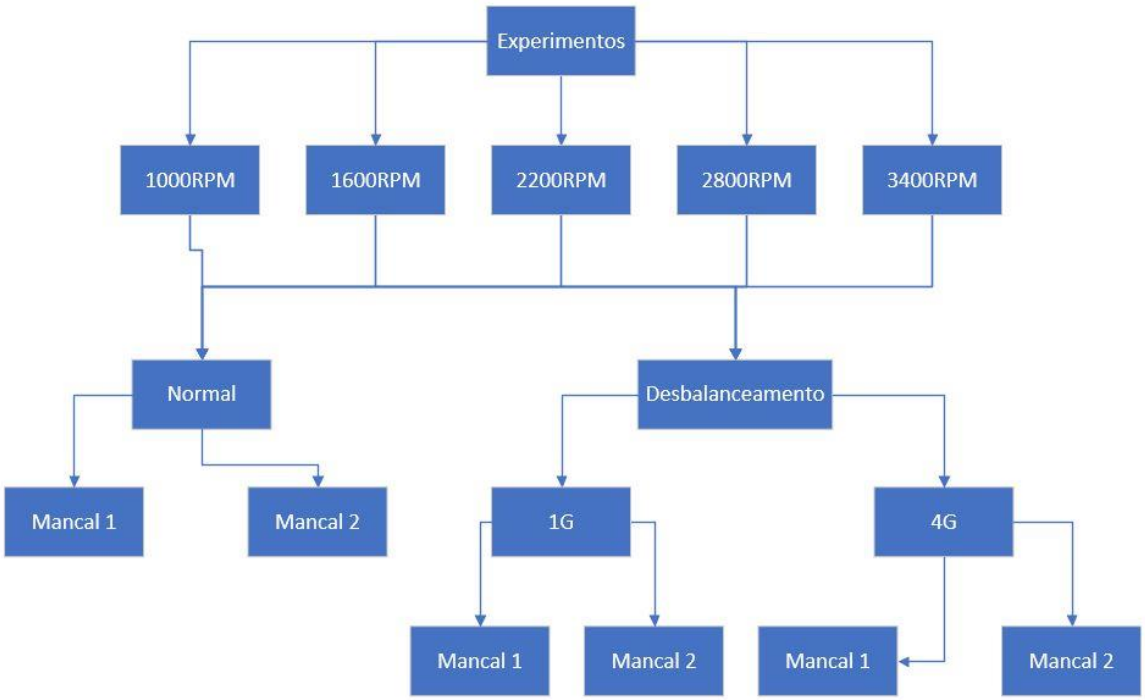
3.3.6 Hyperparameter Tuning

Cada modelo possui parâmetros de treinamento bem diferentes, e que podem alterar a performance do mesmo. Para evitar isso, utilizamos uma técnica chamada de *hyperparameter tuning*, que após avaliar com *cross validation* todas as combinações de parâmetros definidas pelo usuário, indica a que teve melhor resultado.

3.4 EXPERIMENTOS EM LABORATÓRIO

O objetivo da realização dos experimentos em laboratório é validar o modelo de aprendizagem de máquina, obtendo um classificador capaz de identificar, a partir de dados de aceleração obtidos experimentalmente, falhas em máquinas rotativas. Para o desenvolvimento dos experimentos foi primeiramente definido um cronograma, em forma de árvore de experimentos, que define todas as condições simuladas durante esta etapa, como mostra a Figura 21. Para cada configuração foram realizadas de 2 a 3 medições, totalizando 70 experimentos.

FIGURA 21 - Árvore de Experimentos



FONTE: Os Autores, 2021

Para a simulação das falhas foi empregado um Rotor Kit RK 4, da Bently Nevada, ilustrado na Figura 22, com as seguintes especificações:

TABELA 6 - Kit RK 4

Dimensões da base mecânica	165 x 340 x 789 mm
Peso da base mecânica	14,5 Kg
Dimensões do inversor de frequência	115 x 260 x 325 mm
Peso do inversor de frequência	2,7 Kg
Potência	95 a 125 VA (ac), monofásico
Velocidade máxima	10000 RPM

FONTE: Os Autores, 2021

FIGURA 22 - Setup do Rotor Kit

FONTE: Os Autores, 2021

O inversor de frequência presente na Tabela 6 está representado na Figura 23.

FIGURA 23 - Inversor de Frequência

FONTE: Os Autores, 2021

A aquisição de dados se deu por intermédio de um acelerômetro triaxial DeltaTron da Brüel & Kjær, modelo 4535-B-001 com sensibilidade de 96,53 mV/g,

94,06 mV/g e 97,22 mV/g para os eixos X, Y e Z, respectivamente. Adicionalmente, a placa de aquisição utilizada foi o modelo 3160-B-042 da Brüel & Kjær, apresentada na Figura 24. Através do software PULSE LabShop versão 19.0.0.128 - 2014-12-17 foi possível exportar os dados adquiridos em planilhas de Excel, que servem de entrada para o classificador.

Por meio do programa descrito acima pudemos definir a frequência de amostragem dos dados. Esta foi escolhida baseada no método de Nyquist, que sugere utilizar uma frequência de aquisição como sendo no mínimo duas vezes maior que a frequência máxima a ser analisada. Aplicando esse conceito para os nossos experimentos, onde a frequência máxima desejada da FFT corresponde a 6x a frequência de rotação do motor, temos:

$$freq_{max} = 6 * \frac{3400\ RPM}{60} = 340\ Hz$$

$$freq_{nyquist} \geq 2 * 340\ Hz$$

$$freq_{nyquist} \geq 680\ Hz$$

Como a frequência de amostragem padrão do *software* era 1024Hz e atendia a condição descrita acima, optamos por não a alterar.

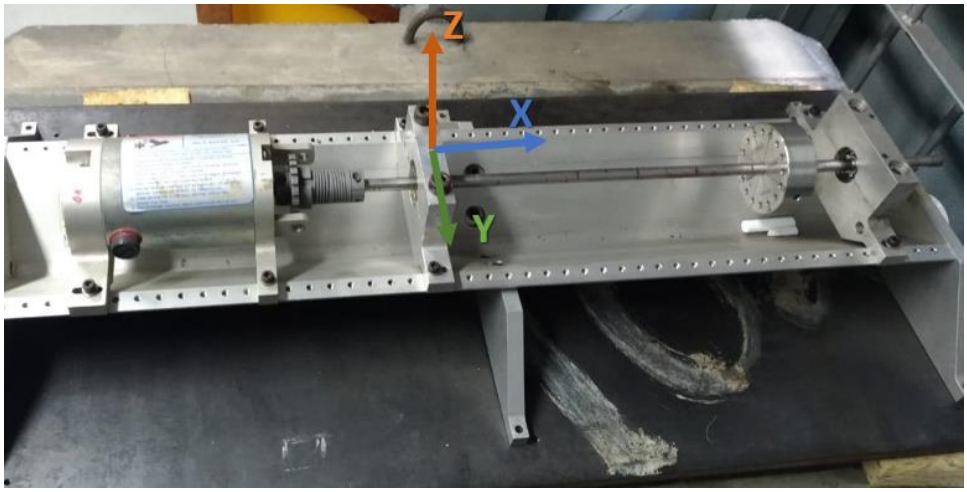
FIGURA 24 - Placa de aquisição de dados

FONTE: Os Autores, 2021

O setup permite a geração de desbalanceamento através da introdução de pesos no disco acoplado ao eixo, o qual se encontra próximo ao mancal mais distante do motor, como pode ser visto no lado direito da Figura 25. Conforme apresentado na árvore de experimentos anteriormente, foram realizados testes com velocidades de rotação de 1000, 1600, 2200, 2800 e 3400 RPM. Além disso, foi simulada a falha de desbalanceamento com pesos de 1 g e 4 g (Figuras 26 e 27), adicionalmente à operação normal. Por fim, o acelerômetro foi posicionado tanto no Mancal 1, mais próximo ao motor, quanto no Mancal 2, mais distante. Essa variação de condições permitiu a obtenção de uma quantidade significativa de dados para classificação e validação do modelo de aprendizagem de máquina treinado com os dados disponíveis no banco de dados MAFAULDA.

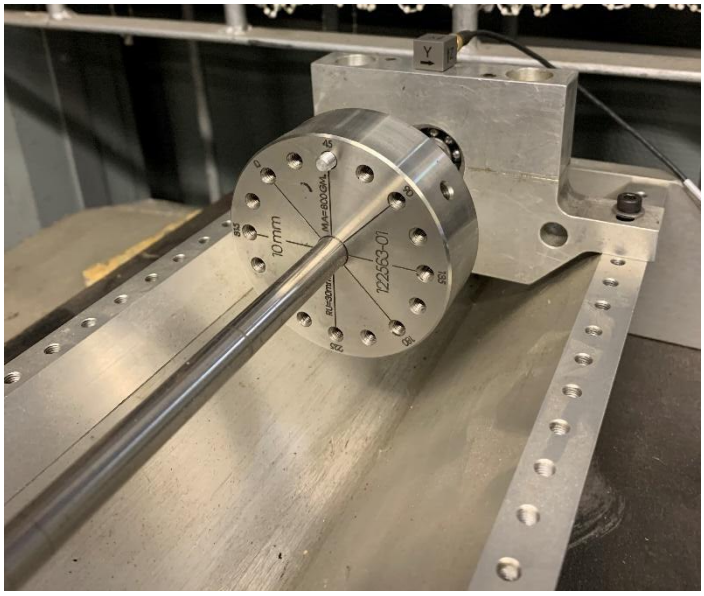
Assim como definido para os dados do MAFAULDA, as direções de medição serão aferidas como X, Y e Z. Elas correspondem às direções axial, transversal e radial respectivamente em relação ao mancal que está posicionado o acelerômetro, como mostra a Figura 25.

FIGURA 25 - Setup Completo



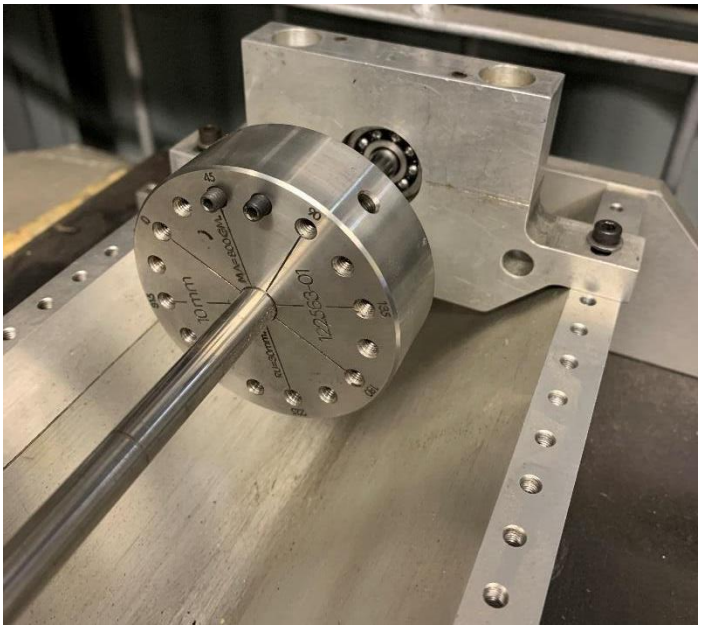
FONTE: Os Autores, 2021

FIGURA 26 - Desbalanceamento 1g



FONTE: Os Autores, 2021

FIGURA 27 - Desbalanceamento 4g



FONTE: Os Autores, 2021

O *setup* descrito acima não possibilita a simulação das falhas de desalinhamento ou de rolamento, portanto estas serão desconsideradas para a geração dos dados experimentais. Estes tipos de falha foram testados com *Cross Validation* para o *dataset* do MAFAULDA.

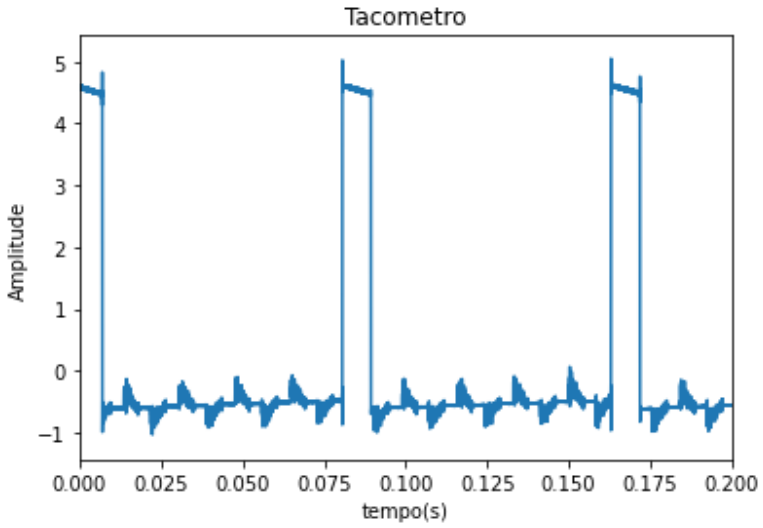
4 RESULTADOS E DISCUSSÕES

Nesse tópico serão apresentados os resultados obtidos após realizarmos os procedimentos discutidos na metodologia, bem como algumas hipóteses e explicações para cada tópico.

4.1 TRATAMENTO E *DATA ANALYSIS* (MAFAULDA)

Com objetivo de conhecer e extrair as melhores informações do banco de dados da UFRJ foram realizadas análises estatísticas das propriedades no domínio do tempo e análises de gráficos de FFTs no domínio da frequência do banco de dados MAFAULDA. O primeiro passo foi avaliar os dados referentes ao tacômetro. Na Figura 28, temos um exemplo do sinal analógico gerado.

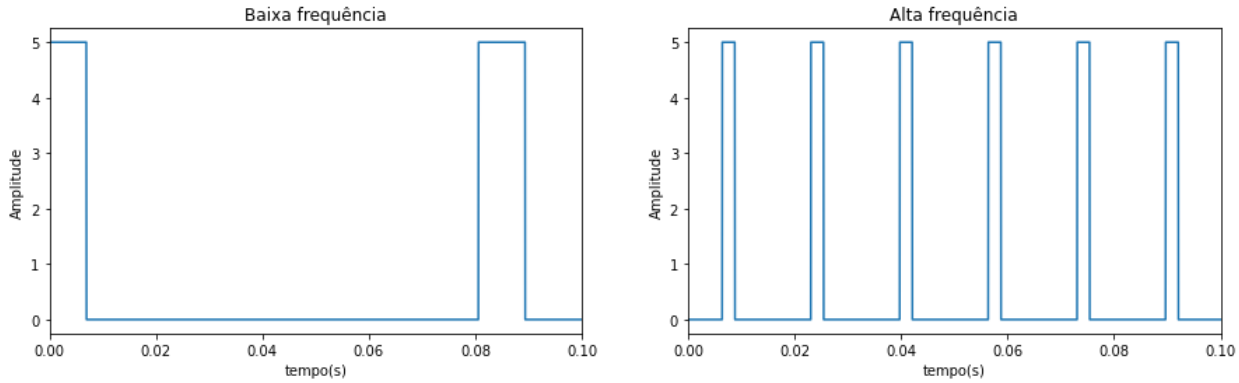
FIGURA 28 - Exemplo de um dos sinais de tacômetro analógico do MAFAULDA



FONTE: Os Autores, 2021

Para facilitar a extração da rotação criamos um filtro digital onde todos os valores foram simplificados para 0 ou 5, como pode ser visto nos exemplos abaixo na Figura 29, para 2 frequências de rotações diferentes.

FIGURA 29 - Sinais do tacômetro filtrados



FONTE: Os Autores, 2021

Posteriormente analisamos os dados extraídos do domínio do tempo, com auxílio de conceitos estatísticos como média, desvio padrão, mínimo, máximo, primeiro, segundo e terceiro quartil para os dados de RMS e pico (peak), como apresentado nas Tabelas 7 e 8 respectivamente.

TABELA 7 - Resumo estatístico do RMS da MAFAULDA

	X1_RMS	Z1_RMS	Y1_RMS	X2_RMS	Z2_RMS	Y2_RMS
média	0,54	0,47	0,29	0,21	0,03	0,49
desvio padrão	0,19	0,46	0,19	0,20	0,01	0,40
min	0,28	0,09	0,03	0,03	0,02	0,06
25%	0,43	0,19	0,14	0,07	0,03	0,27
50%	0,49	0,31	0,27	0,12	0,03	0,34
75%	0,61	0,52	0,43	0,30	0,03	0,49
max	1,82	2,52	0,82	1,40	0,12	2,05

FONTE: Os Autores, 2021

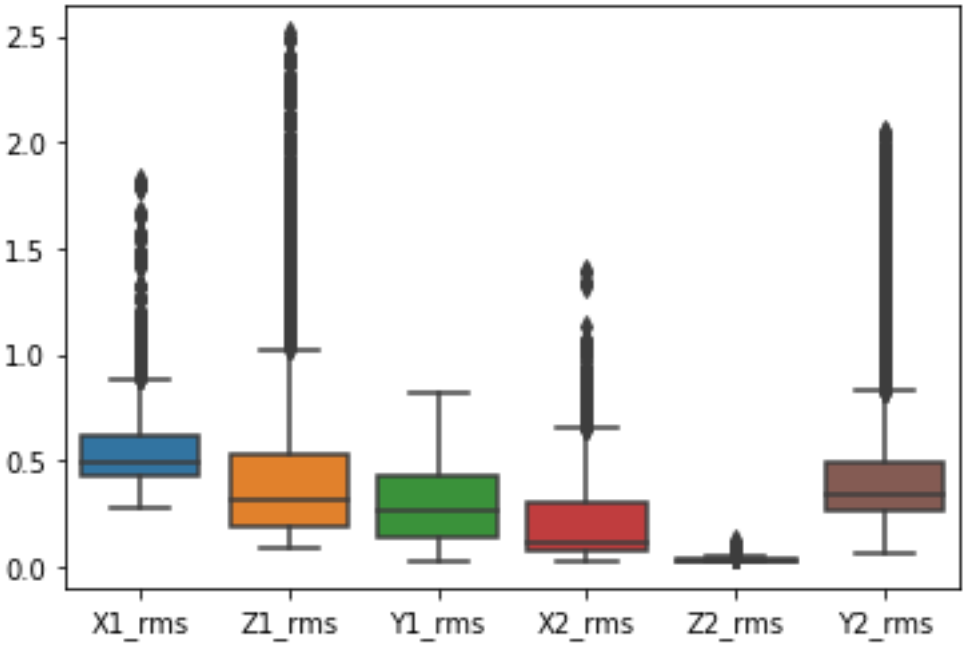
TABELA 8 - Resumo estatístico do pico da MAFAULDA

	X1_peak	Z1_peak	Y1_peak	X2_peak	Z2_peak	Y2_peak
média	2,06	1,98	1,34	0,74	0,11	1,58
desvio padrão	0,70	1,58	0,86	0,71	0,04	0,78
min	0,97	0,36	0,12	0,11	0,07	0,25
25%	1,65	0,90	0,57	0,26	0,09	1,10
50%	1,98	1,47	1,24	0,41	0,10	1,41
75%	2,37	2,45	1,93	1,11	0,12	1,76
max	19,60	9,11	4,59	4,79	0,57	5,87

FONTE: Os Autores, 2021

À primeira vista, os valores pareciam estar bem distribuídos, com uma vibração maior na direção X para o acelerômetro 1 (mais próximo do motor) e, na direção Y, a vibração se intensificando para o acelerômetro 2. Porém podemos notar um comportamento estranho dos dados obtidos na direção Z2, onde fica evidente que os valores estão fora da média. Para facilitar a visualização da distribuição dos dados criamos um diagrama de caixa para os valores de RMS, visto na Figura 30.

FIGURA 30 - Diagrama da caixa para os valores de RMS(g)

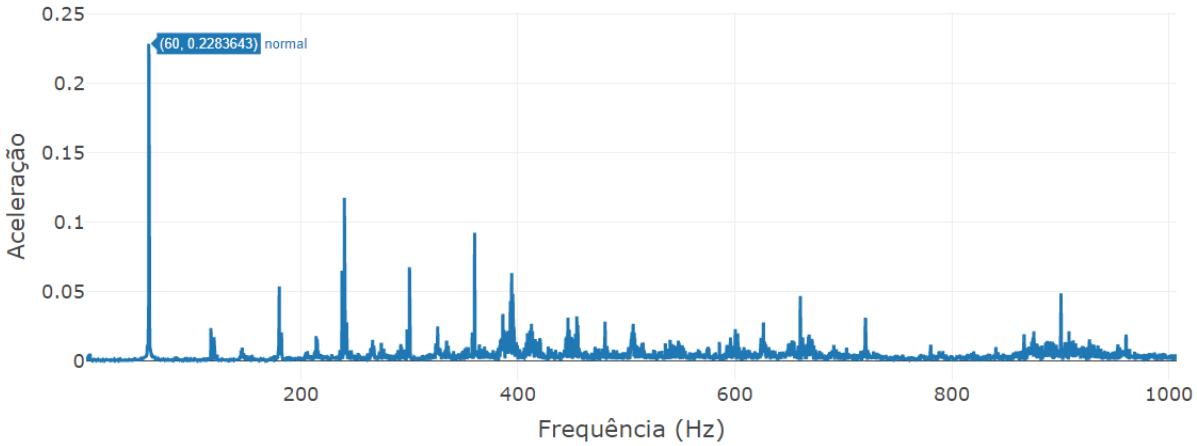


FONTE: Os Autores, 2021

A partir dessa imagem fica claro que os valores correspondentes a direção Z no acelerômetro 2 estão fora do esperado, com valores muito baixos. Com essa constatação tivemos que limitar o banco de dados as direções X e Y para não comprometer o modelo de aprendizado de máquina com dados incertos.

Dando sequência à análise dos dados, examinamos a FFT gerada a partir dos dados de tempo, tal como exposto Figura 31. Os resultados obtidos foram bem satisfatórios quando observamos as frequências baixas (0 -1000Hz), onde ficam claros picos de vibração na frequência de rotação do motor.

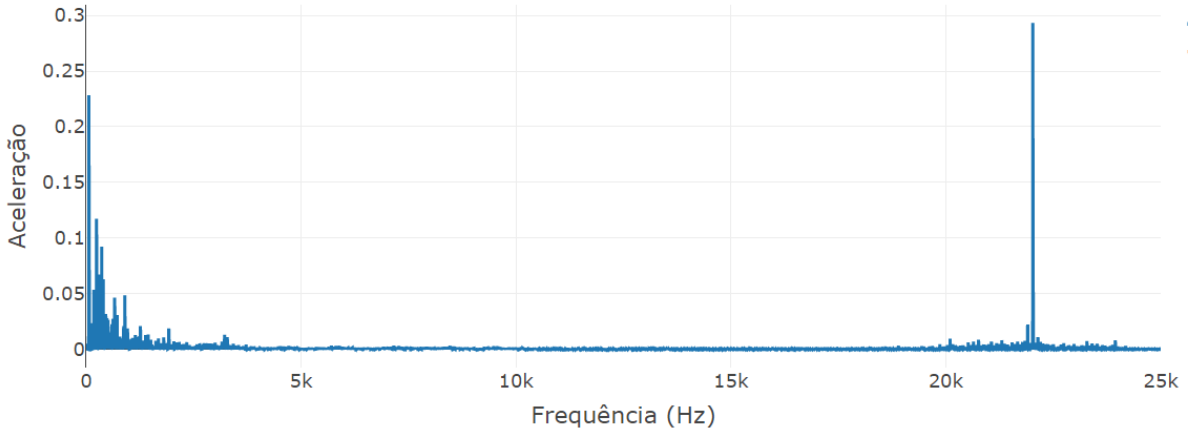
FIGURA 31 - FFT [0-1kHz] direção Z1 (radial, acelerômetro 1) para uma rotação do motor de 60Hz



FONTE: Os Autores, 2021

Contudo ao analisar o espectro completo, mostrado na Figura 32, é possível observar um pico inesperado em uma frequência próxima de 22kHz, que se repete para todos os experimentos. Como a faixa de medição dos acelerômetros mais alta é 10kHz acreditamos que esse pico não corresponde a uma medição correta.

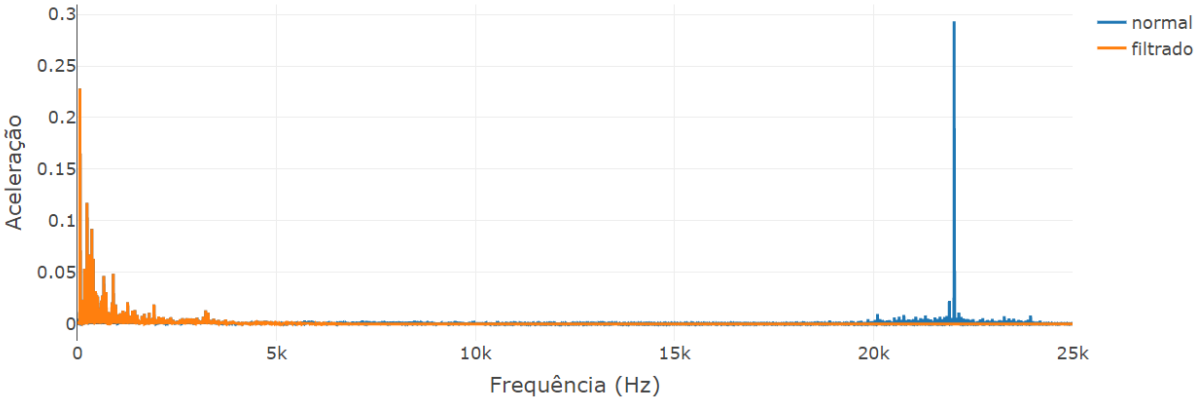
FIGURA 32 - FFT completo, direção Z1 (radial, acelerômetro 1)



FONTE: Os Autores, 2021

Portanto para corrigir esse problema resolvemos utilizar um filtro digital passa baixa de 10ª ordem em 5kHz, já que esse era o alcance máximo de um dos acelerômetros utilizados e as frequências mais altas não são relevantes para este trabalho. A comparação do antes e depois do filtro pode ser observada na Figura 33, com o sinal filtrado representado em laranja.

FIGURA 33 - FFT do sinal filtrado em comparação ao original



FONTE: Os Autores, 2021

Para avaliar a interferência desse pico inesperado comparamos as médias dos valores com e sem filtro Tabela 9.

TABELA 9 - Comparação das propriedades extraídas no domínio do tempo com e sem a presença de um filtro

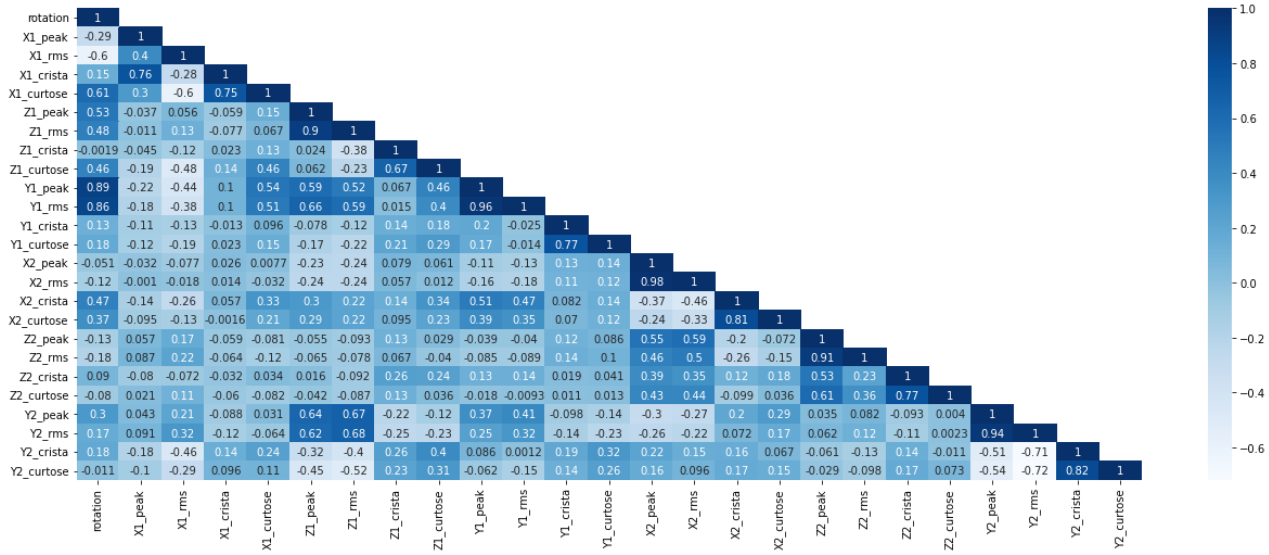
	Média		
	Sem filtro	**Com Filtro**	**var %**
crista	3,99	4,04	1,12
curtose	2,78	3,03	8,89
pico	1,85	1,30	29,74
RMS	0,48	0,34	29,49

FONTE: Os Autores, 2021

Apesar dos valores de crista não serem tão influenciados podemos observar uma variação significativa dos valores de pico e RMS.

Voltando para as propriedades no domínio do tempo optamos por utilizar a matriz de correlação de Pearson, visto na Figura 34, para detectar a correlação das nossas *features* extraídas.

FIGURA 34 - Matriz de correlação de Pearson para as propriedades extraídas do domínio do tempo



FONTE: Os Autores, 2021

Em decorrência, decidimos por retirar as estatísticas relacionadas ao pico, já que estas apresentavam uma forte correlação (acima de 0,85 na escala de Pearson) com outras propriedades, o que não é recomendado para alguns modelos de aprendizado de máquina.

4.2 TREINAMENTO DOS MODELOS DE APRENDIZADO DE MÁQUINA

Definidas as *features* e com as etapas de tratamento de dados concluídas, foram empregados os 4 modelos de aprendizagem de máquina para treinamento definidos na seção de Metodologia: SVM, LGBM *Classifier*, *Random Forest Classifier* e MLP. Com o auxílio dos algoritmos disponíveis nos Apêndices 3 e 4, foi possível treinar a máquina para cada um destes modelos, obter as respectivas acurácias e realizar *hyperparameter tuning* dos parâmetros. A Tabela 10 traz os resultados de acurácia obtidos antes do emprego do *tuning*.

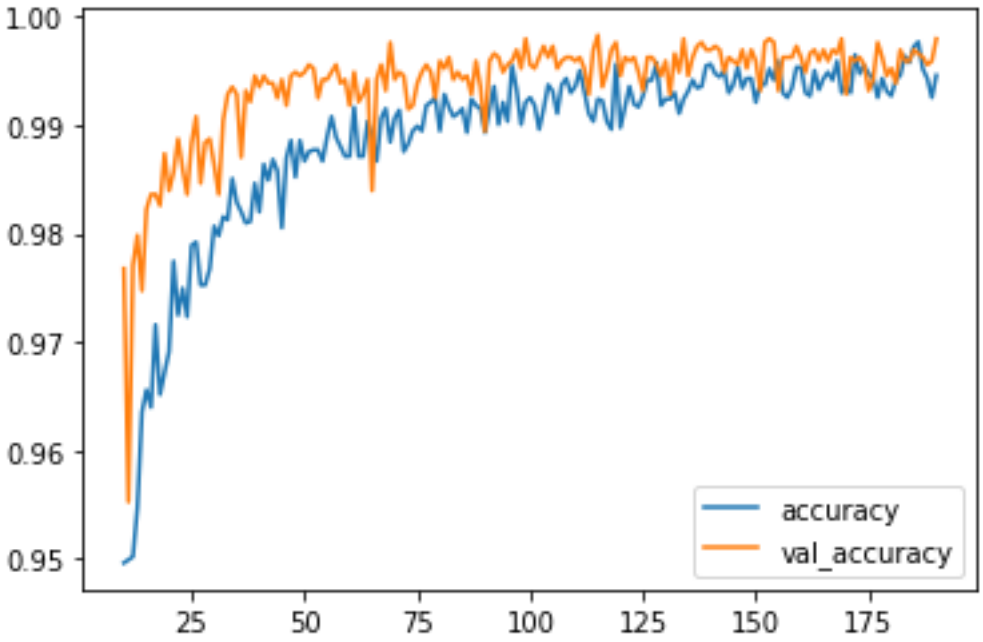
TABELA 10 - Acurácia dos modelos para *train-test-split*

Modelo	**Acurácia**
SVM	92,76%
LGBM Classifier	99,76%
Random Forest	99,95%
MLP	99,66%

FONTE: Os Autores, 2021

O Gráfico 1 a seguir apresenta a curva de aprendizagem para o MLP.

GRÁFICO 1 - Curva de aprendizagem MLP



FONTE: Os Autores

Estes valores foram obtidos através da separação dos dados com 70% para treinamento e 30% para teste. Os parâmetros das máquinas não foram alterados nesta etapa, exceto para o MLP, ou seja, os valores padrão foram utilizados. A não alteração dos valores se baseou no fato de que os valores ótimos não eram conhecidos e serão testados posteriormente com a etapa de *tuning.* Estes podem ser observados nas Tabelas 11, 12, 13 e 14.

TABELA 11 - Parâmetros de análise para SVM

Support Vector Machine	
Cost	1
Degree	3

FONTE: Os Autores, 2021

TABELA 12 - Parâmetros de análise para LGBM Classifier

Light Gradient Boosting Machine	
Num Leaves	31
Max Depth	-1

FONTE: Os Autores, 2021

TABELA 13 - Parâmetros de análise para Random Forest

Random Forest	
Bootstrap	True
Max Depth	None
Max Features	Auto
Min Samples Leaf	1
Min Samples Split	2
N Estimators	100

FONTE: Os Autores, 2021

TABELA 14 - Parâmetros de análise para o MLP

Batch Size	64
Drop Rate	0.5
Neurons	[100,50,50]
Activation	Softmax e Relu
Optimizer	Adam

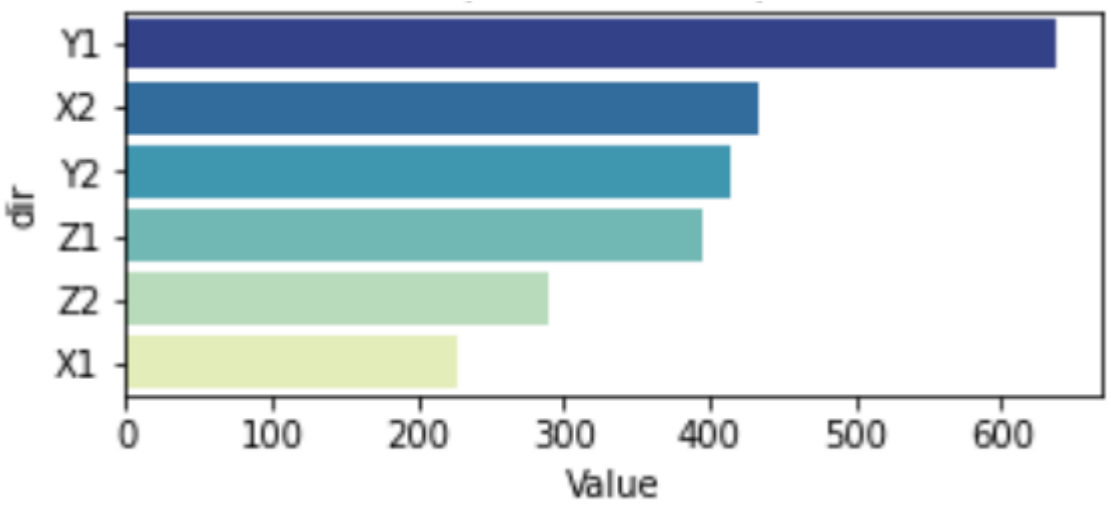
FONTE: Os Autores, 2021

A etapa de *tuning* vai emprega *Cross Validation*, dividindo o *dataset* em 5 partes iguais e prosseguindo conforme exemplificado na seção 3.3.5.2 da metodologia. Tendo em vista que os diferentes modelos de aprendizado de máquina envolvidos contêm diferentes parâmetros de controle, os valores considerados e os resultados obtidos para o *tuning* são apresentados. Entretanto, definições mais precisas das funções desempenhadas por cada um deles dentro do modelo não serão abordadas.

Os parâmetros avaliados no *tuning* do modelo SVM foram *cost* e *degree*, variando de 1 a 10 e de 2 a 6, respectivamente. O resultado considerado ótimo foi obtido para um valor de *9* para o *cost* e para um valor de 2 de *degree*. Neste caso, a acurácia obtida foi 98,98%.

Abordando o modelo LGBM Classifier, foram avaliados os valores [-1, 2, 3, 4, 5, 6] para a *max depth* e os valores [7, 20, 30, 40, 50, 60] para *num leaves*. O aumento de acurácia, neste caso, não foi tão significativo, indo de 99,76% para 99,84%. Este modelo em específico permite, além da avaliação da acurácia, uma abordagem da importância das *features* selecionadas para o treinamento da máquina. A Figura 35 apresenta o resultado das *features* compilados para cada direção.

FIGURA 35 - Importância das direções para o modelo LGBM



FONTE: Os Autores

É possível observar que o eixo Z do Mancal 2 apresenta a menor influência ao modelo quando comparada com os outros eixos. Também é importante citar o peso atrelado ao eixo Y do Mancal 1, fato que será levado em consideração em análises posteriores.

Dando continuidade à análise de otimização dos modelos, a Tabela 15 contém os parâmetros avaliados para o *Random Forest Classifier*, bem como seus respectivos valores.

TABELA 15 - Parâmetros de tuning do *Random Forest Classifier*

Bootstrap	True, False
Max Depth	5, 10, 20, 30
Max Features	Auto, Sqrt
Min Samples Leaf	1, 2, 4
Min Samples Split	2, 5, 7, 10
N Estimators	50, 100, 200, 300, 400, 500

FONTE: Os Autores, 2021

O ponto considerado ótimo, para este caso atrelado a uma acurácia de 100%, foi encontrado para valores: False, 30, Auto, 1, 2 e 200, conforme ordem apresentada na Tabela 15. Para o último modelo empregado, MLP, os parâmetros e seus respectivos valores estão dispostos na Tabela 16.

TABELA 16 - Parâmetros de tuning para o MLP

Batch Size	16, 32, 64, 128
Drop Rate	0.1, 0.3, 0.5, 0.7, 0.9
Loss	Categorical Crossentropy, Poisson, Categorical Hinge
Activation	Softmax, Relu, Sigmoid, Tanh
Optimizer	SGD, Adam

FONTE: Os Autores, 2021

Os valores testados para o parâmetro *neurons* foram todas as possíveis combinações dos números 100, 50 e 30. Por exemplo: [100, 30, 100], [50, 30, 100] e assim por diante. São avaliados, neste caso, os valores ótimos de neurônios na camada de entrada e nas duas camadas subsequentes. O resultado obtido para a rede neural MLP está disposto na Tabela 17 e pode ser simulado com o código disponível no Apêndice 5.

TABELA 17 - Melhores parâmetros encontrados

Batch Size	32
Drop Rate	0.3
Neurons	[100, 50, 50]
Loss	Categorical Crossentropy
Activation	Softmax
Optimizer	Adam

FONTE: Os Autores, 2021

Estes valores permitiram o modelo alcançar uma acurácia de 99,83%.

O código em *Python* desenvolvido para realizar esta análise está disponível nos Apêndices 3, 4 e 5.

4.3 ANÁLISE DA INFLUÊNCIA DOS DADOS NOS MODELOS DE APRENDIZADO DE MÁQUINA

Para os resultados obtidos acima foram utilizados todos os dados extraídos segundo a metodologia, no total 9755 dados. Porém, com o objetivo de avaliar a influência da quantidade dos dados na precisão do modelo de aprendizado de máquina escolhemos os modelos com melhores performances (*random forest classifier* e *LGBM classifier) e* separamos o conjunto de dados em 80%- 20% para treinamento - validação respectivamente. Entre os dados de treinamento selecionamos amostras segundo a Tabela 18 e verificamos a *accuracy* obtida para classificar os dados de validação. Os dados podem ser observados nas Tabelas 18 e 19 a seguir.

TABELA 18 - Acurácia vs Quantidade de dados *Random Forest*

Amostra (%)	**Quantidade de dados**	**Acurácia (%)**
1	78	70,63
3	234	85,08
5	390	89,80
10	780	96,10
20	1561	98,36
30	2341	99,59
40	3122	99,79
50	3902	99,44
60	4682	99,69
70	5463	99,90
80	6243	100
90	7024	100
100	7804	100

FONTE: Os Autores, 2021

TABELA 19 - Acurácia vs Quantidade de dados LGBM

Amostra (%)	Quantidade de dados	Acurácia (100%)
1	78	62,69
3	234	78,99
5	390	87,70
10	780	93,39
20	1561	97,54
30	2341	98,97
40	3122	98,92
50	3902	99,59
60	4682	99,49
70	5463	99,85
80	6243	99,90
90	7024	99,90
100	7804	99,95

FONTE: Os Autores, 2021

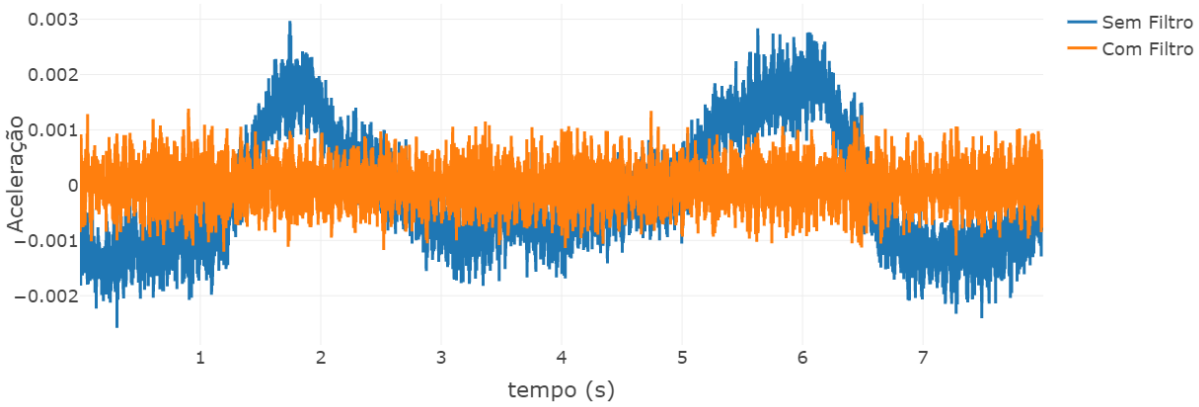
Como pode ser visto nas tabelas acima, embora os resultados continuem satisfatórios com a redução dos dados, se mostrou acertada a decisão de aumentar em 5 vezes a quantidade de dados. Ainda podemos observar como o modelo *Random Forest Classifier* comporta-se melhor em praticamente todas as amostras, comprovando assim a fundamentação teórica que não recomenda o uso do LGBM para poucos dados, fator determinante para a escolha do classificador final.

4.4 EXPERIMENTOS

Dentre as várias opções contidas no software empregado durante a realização dos experimentos, foram geradas planilhas em Excel contendo os dados de aceleração adquiridos pelo acelerômetro, bem como as FFTs dos sinais. Através do Apêndice 6 foi possível extrair os dados da planilha, tratá-los e gerar os gráficos de Aceleração x Tempo e as FFTs correspondentes, que são mostrados nos Gráficos de 2 a 19.

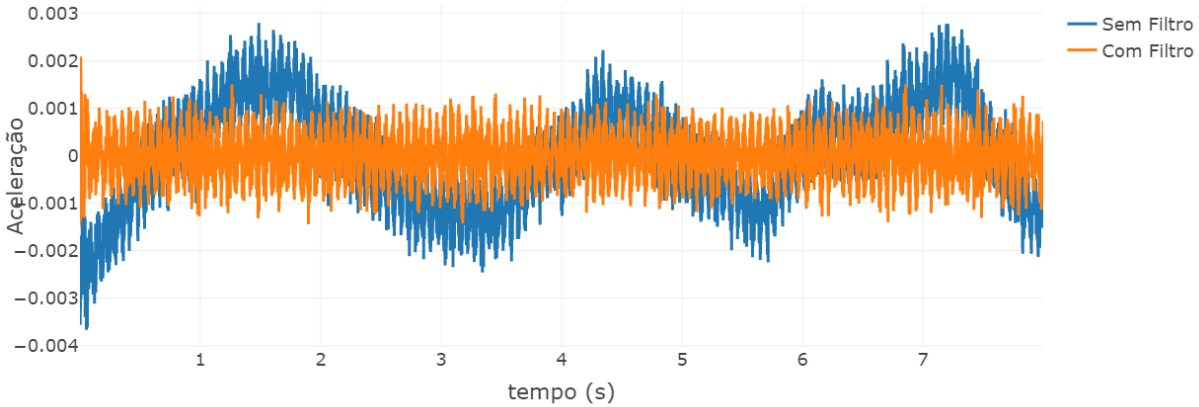
Devido a grande quantidade de experimentos conduzidos, serão expostos apenas parte dos gráficos obtidos. Além disso, é importante citar que apenas os resultados para o eixo Y estão contemplados. O motivo é, conforme apresentado anteriormente, a importância deste eixo para o treinamento das máquinas, evidenciado na Figura 33.

GRÁFICO 2 - Normal (1000RPM) no tempo



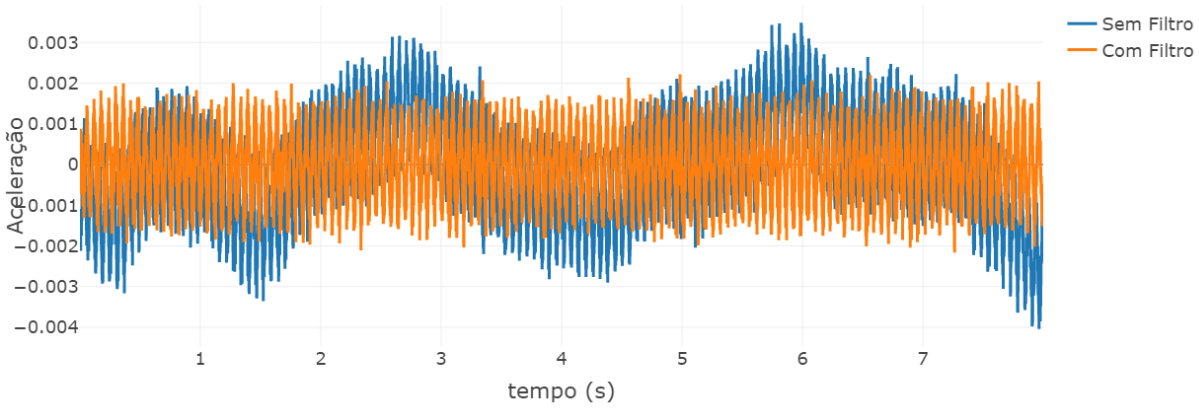
FONTE: Os Autores, 2021

GRÁFICO 3 - Desbalanceado 1g (1000RPM) no tempo



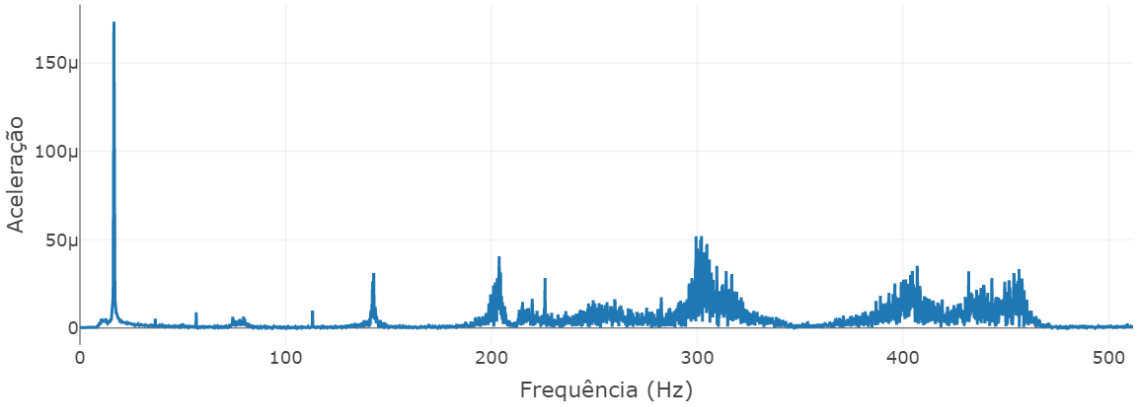
FONTE: Os Autores, 2021

GRÁFICO 4 - Desbalanceado 4g (1000RPM) no tempo



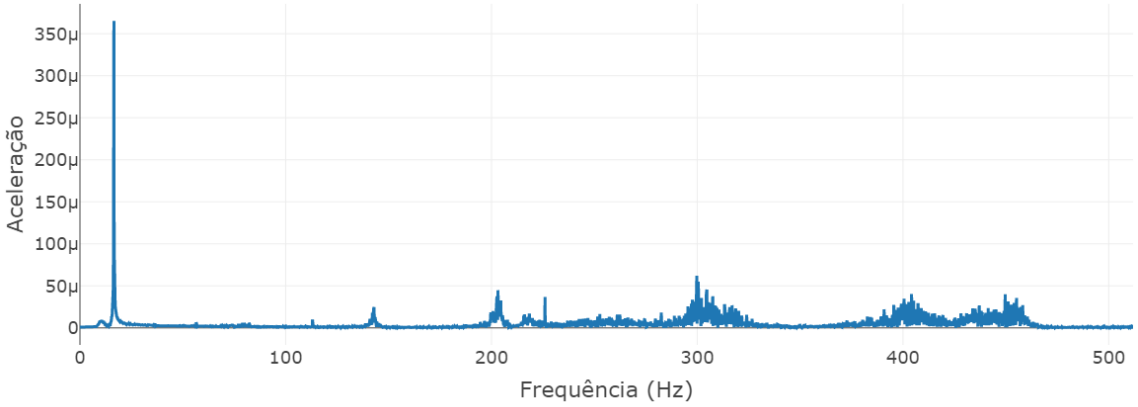
FONTE: Os Autores, 2021

GRÁFICO 5 - FFT Normal (1000RPM)



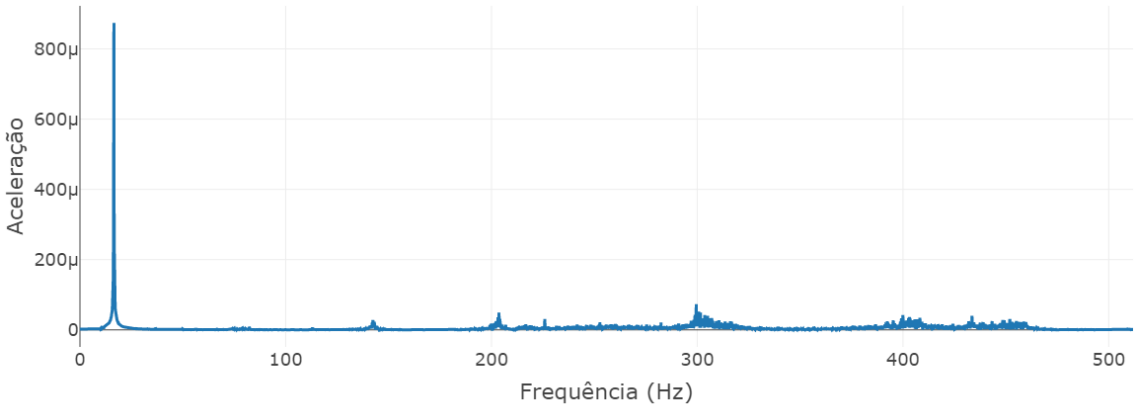
FONTE: Os Autores, 2021

GRÁFICO 6 - FFT Desbalanceado 1g (1000RPM)



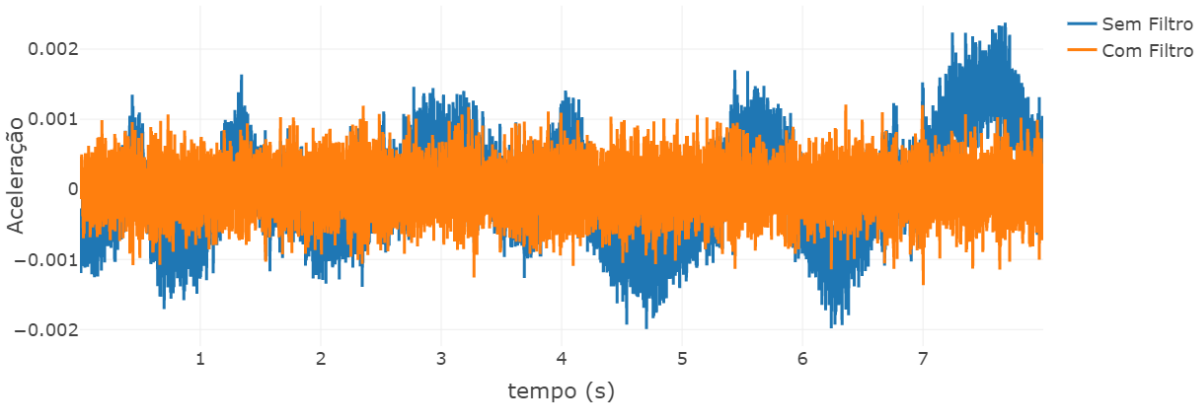
FONTE: Os Autores, 2021

GRÁFICO 7 - FFT Desbalanceado 4g (1000RPM)



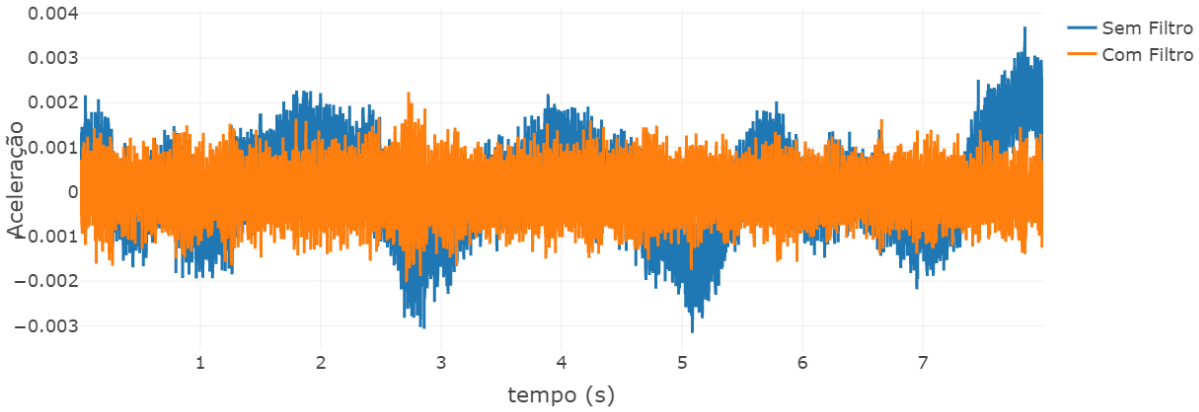
FONTE: Os Autores, 2021

GRÁFICO 8 - Normal (2200RPM) no tempo



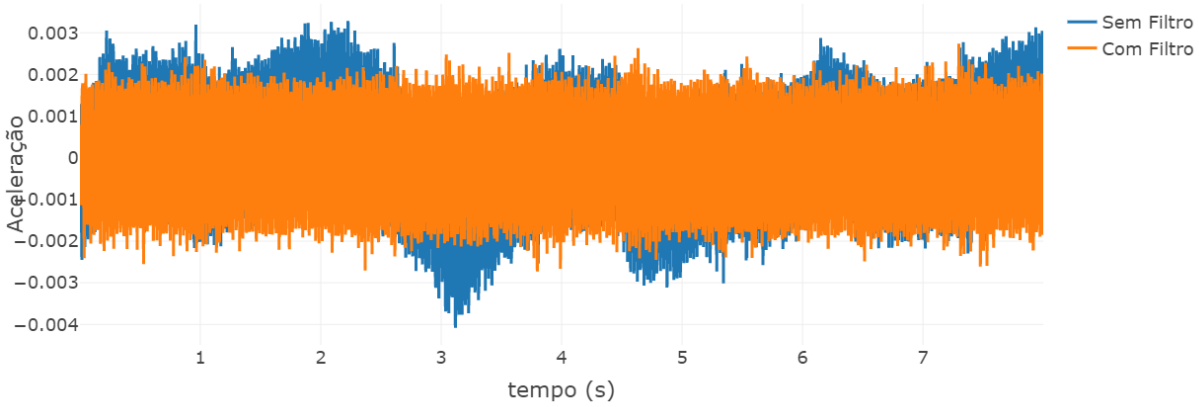
FONTE: Os Autores, 2021

GRÁFICO 9 - Desbalanceado 1g (2200RPM) no tempo



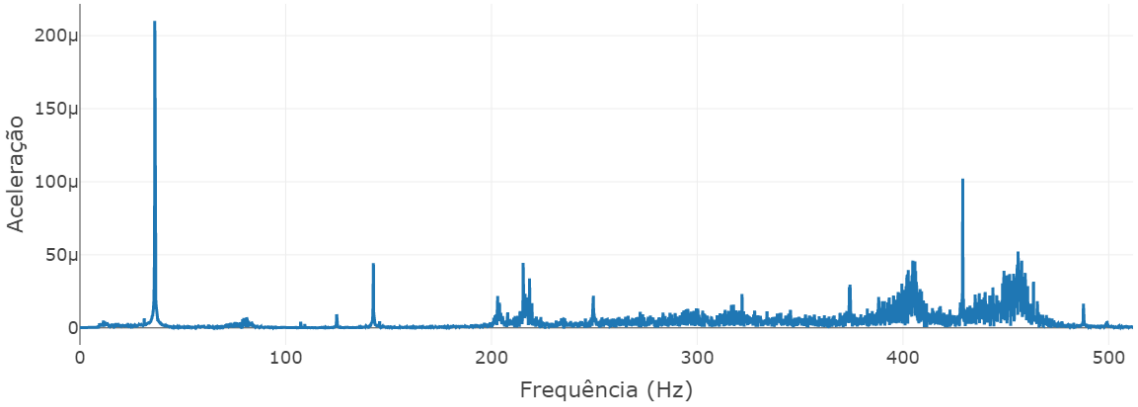
FONTE: Os Autores, 2021

GRÁFICO 10 - Desbalanceado 4g (2200RPM) no tempo



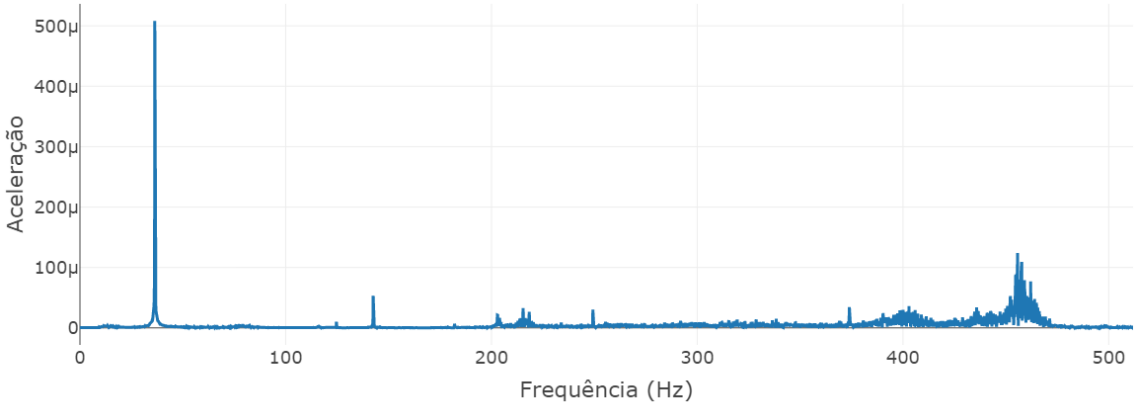
FONTE: Os Autores, 2021

GRÁFICO 11 - FFT Normal (2200RPM)



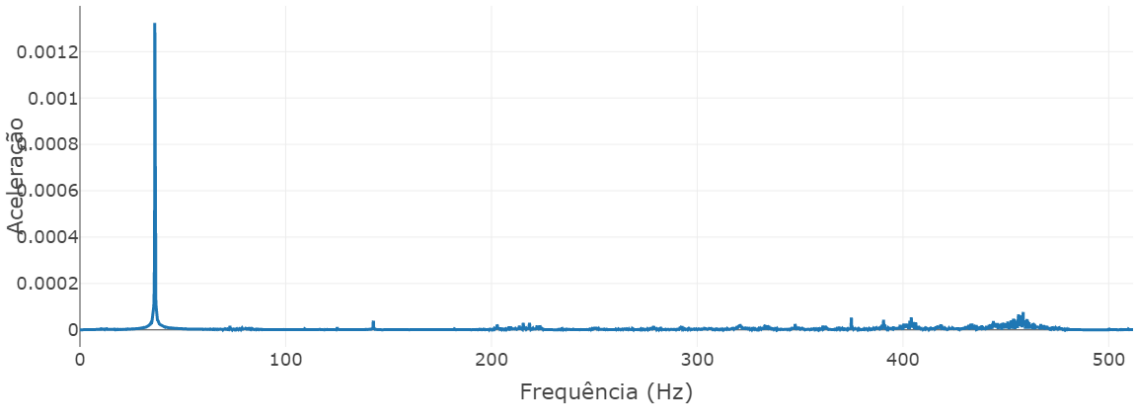
FONTE: Os Autores, 2021

GRÁFICO 12 - FFT Desbalanceado 1g (2200RPM)



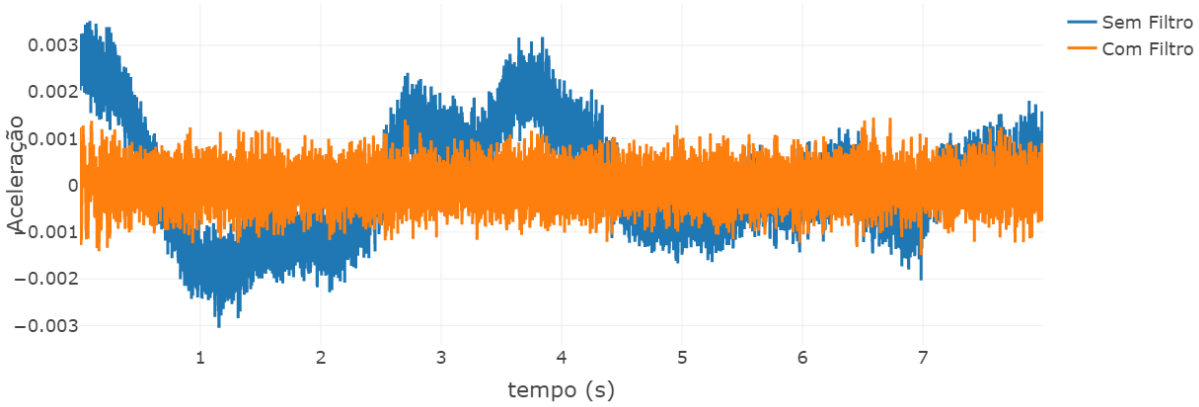
FONTE: Os Autores, 2021

GRÁFICO 13 - FFT Desbalanceado 4g (2200RPM)



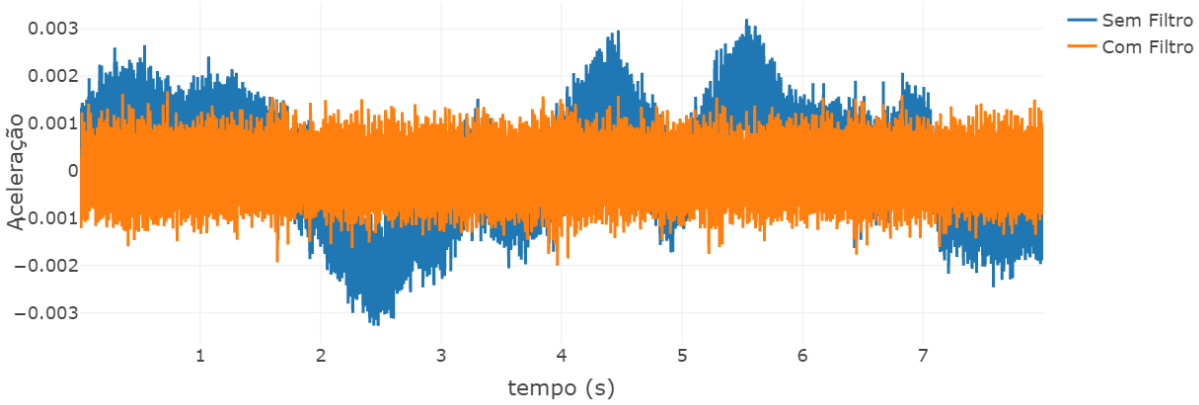
FONTE: Os Autores, 2021

GRÁFICO 14 - Normal (3400RPM) no tempo



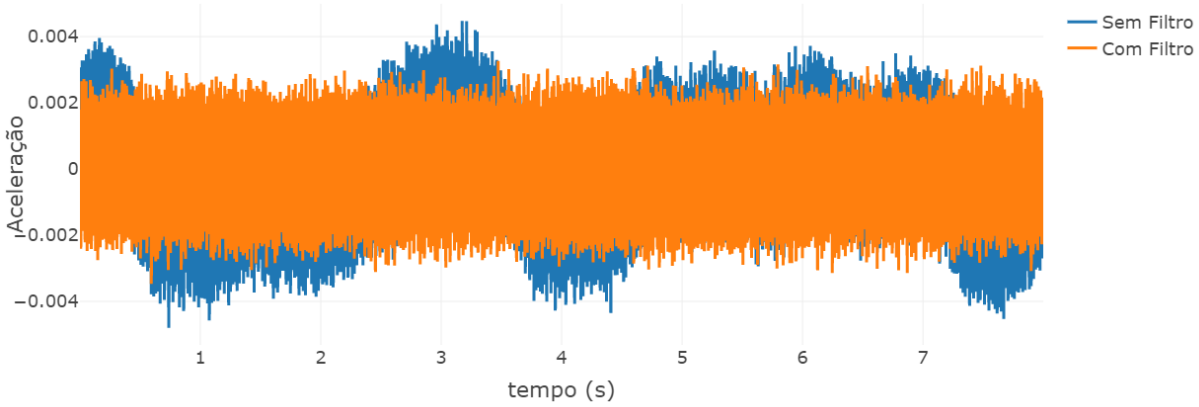
FONTE: Os Autores, 2021

GRÁFICO 15 - Desbalanceado 1g (3400RPM) no tempo



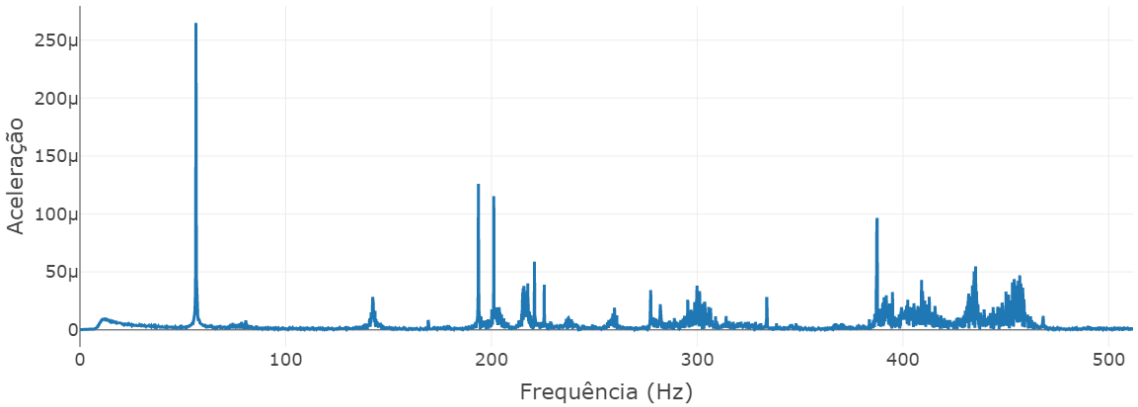
FONTE: Os Autores, 2021

GRÁFICO 16 - Desbalanceado 4g (3400RPM) no tempo



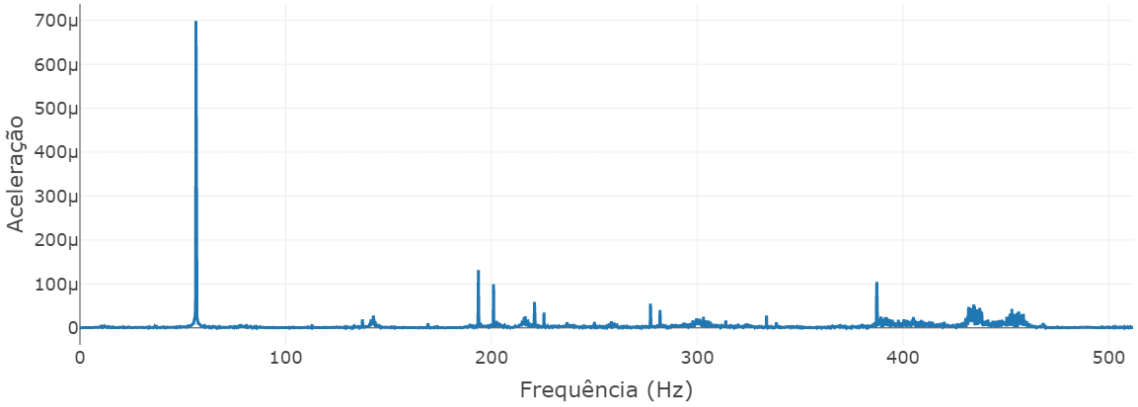
FONTE: Os Autores, 2021

GRÁFICO 17 - FFT Normal (3400RPM)



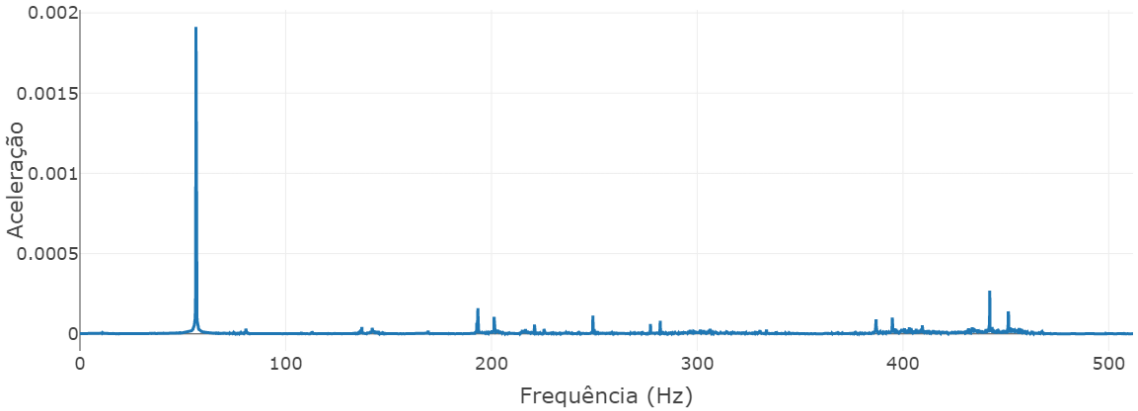
FONTE: Os Autores, 2021

GRÁFICO 18 - FFT Desbalanceado 1g (3400RPM)



FONTE: Os Autores, 2021

GRÁFICO 19 - FFT Desbalanceado 4g (3400RPM)



FONTE: Os Autores, 2021

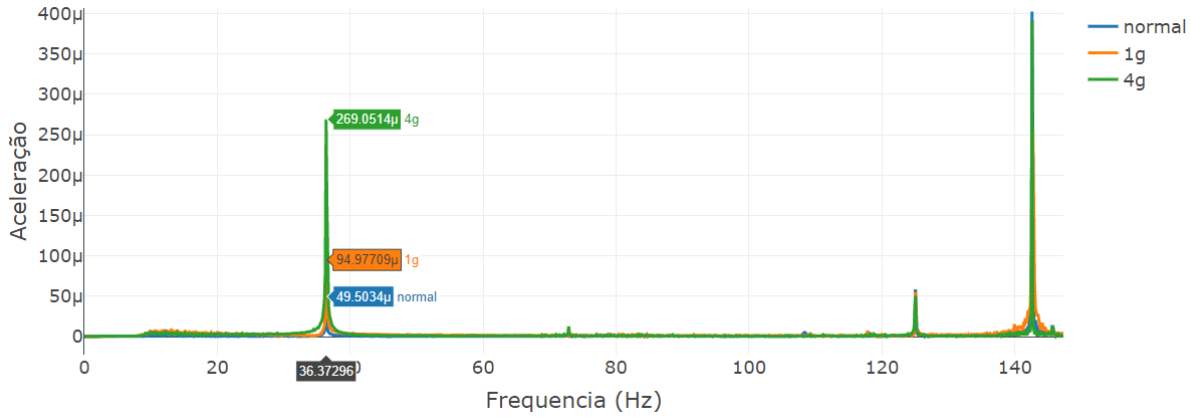
Vale ressaltar que o algoritmo foi programado para ajustar a escala dos gráficos automaticamente. Como as FFTs para o desbalanceamento apresentam amplitudes maiores no primeiro pico, tem-se a impressão de um sinal mais "limpo", o que na verdade não ocorre.

Como se pode observar nos gráficos apresentados para Aceleração x Tempo (Gráficos 2, 3, 4, 8, 9, 10, 14, 15 e 16) o sinal sem filtro não condiz com o esperado para uma máquina rotativa, visto que há o aparecimento de um pico muito próximo de 0 Hz. Empregando-se um filtro passa alta de ordem 10 na frequência de 10 Hz, foi possível obter o sinal filtrado apresentado nos gráficos na cor laranja.

Foram levantadas duas hipóteses para explicar o comportamento da aceleração no sinal não filtrado. A primeira delas é a existência de um modo de corpo rígido no sistema devido à dificuldade de restringir por completo a base mecânica. Alternativamente, considera-se falha no acelerômetro em identificar baixas frequências. De qualquer forma, as frequências filtradas não estão relacionadas as falhas estudadas neste trabalho e, portanto, não teriam influência nos resultados finais, sendo, desta forma, retiradas do sinal.

Os gráficos de FFT (Gráficos 5, 6, 7, 11, 12, 13, 17, 18 e 19) mostram claramente aumento significativo na amplitude do primeiro pico, cuja frequência corresponde a frequência de rotação da máquina, quando o sistema apresenta desbalanceamento, o que é previsto na bibliografia. Nos experimentos para rotação de 2200 RPM para a direção Z-radial, por exemplo, observa-se um aumento de um valor próximo a 50 microns para amplitude superior a 90 microns, no caso de desbalanceamento com 1 grama, e para amplitude próxima a 270 microns no caso de 4 gramas. Este último representando um aumento superior a 5x o valor do primeiro pico, conforme Figura 36.

FIGURA 36 - FFT compilada para os experimentos

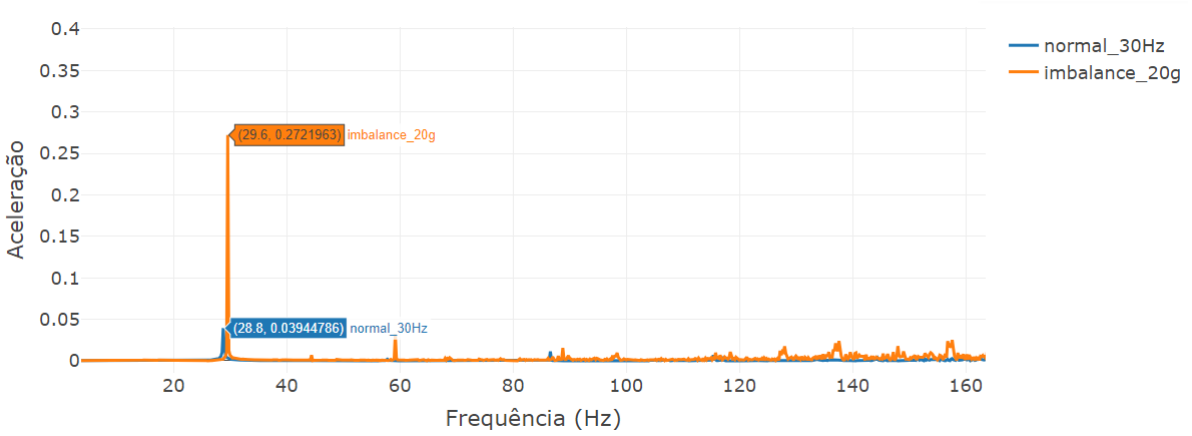


FONTE: Os Autores, 2021

4.5 COMPARAÇÃO MAFAULDA VS EXPERIMENTO

Assim como nos experimentos também é possível identificar claramente um aumento na frequência de 1x a rotação. Na Figura 37 temos em laranja a FFT para um caso de desbalanceamento, enquanto em azul está representado o caso normal.

FIGURA 37 - FFT compilada para o MAFAULDA



FONTE: Os Autores, 2021

Apesar da semelhança no comportamento das frequências, é possível notar que os níveis de vibração estão em grandezas diferentes. Analisando as amplitudes de vibração para as Figuras 34 e 35 em condições normais, temos uma amplitude de 0,0395 para o MAFAULDA enquanto para os experimentos em condições semelhantes de operação o pico foi de 49 microns. Comprova-se assim, o que foi apresentado pela ISO 10816 (Figura 5), na qual os níveis de vibração variam consideravelmente conforme o tamanho da máquina. Ainda analisamos essa diferença para as propriedades obtidas no domínio do tempo, e o resultado pode ser observado na Tabela 20 abaixo.

TABELA 20 - Comparação MAFAULDA vs Experimentos no domínio do tempo

	Média		
	MAFAULDA	**Experimentos**	**var %**
crista	4,035	3,560	11,768
curtose	3,030	2,563	15,403
pico	1,303	0,002	99,829
RMS	0,339	0,001	99,803

FONTE: Os Autores, 2021

Tendo em vista que os dados não estavam nivelados seria inviável tentar classificá-los a partir dos dados do MAFAULDA. Para resolver esse problema tivemos que alterar as *features* do banco de dados e dos experimentos de forma a generalizar

o modelo de aprendizado de máquina para detectar padrões de comportamento, assim como foi apresentado nas Figuras 36 e 37, e não níveis de vibração.

4.6 SELEÇÃO DE *FEATURES* E CLASSIFICADOR FINAL

Para a conclusão da última etapa deste estudo, foi realizado um trabalho de filtragem e seleção das *features* disponíveis para que o algoritmo que serve como classificador final pudesse realizar sua tarefa para diversas configurações de máquinas rotativas, independentemente de suas dimensões e demais características. Isso foi necessário devido ao fato de que os dados disponibilizados pelo MAFAULDA são muito específicos para o caso analisado e estavam gerando resultados insatisfatórios, com precisão de 66,67% para a classificação dos experimentos realizados em laboratório.

Como o *setup* experimental permitia somente a simulação da falha de desbalanceamento e, segundo a bibliografia, esta se apresenta predominantemente na amplitude dos primeiros e segundos picos, retiramos todas as *features* relacionadas aos picos subsequentes, para os eixos X e Y. Esta modificação, aliada às duas filtragens ocorridas após etapa de *Data Analysis* do *dataset* do MAFAULDA, em que foram eliminadas as *features* relacionadas ao pico e ao eixo Z, a quantidade inicial de 61 foi reduzida para 11.

A comparação entre os gráficos das FFTs obtidas com os dados do MAFAULDA e os extraídos dos experimentos em laboratório constataram discrepâncias significativas nos picos. Para contornar este problema, que impedia a generalização completa do sistema, dividimos os valores dos picos pelo valor do RMS do eixo correspondente, obtendo assim 9 *features*: rotação, valores de crista e curtose para os eixos X e Y e os valores dos primeiros e segundos picos dos eixos divididos pelos seus respectivos valores de RMS. Isso foi realizado com auxílio do código apresentado no Apêndice 7.

Embora pequena a diferença de acurácia obtida para o modelo *Random Forest Classifier* em relações às outras máquinas, este foi o motivo principal para sua escolha na classificação dos experimentos. Além disso, levou-se em consideração sua boa aplicabilidade a atividades de classificação e o fato de manter a acurácia mesmo quando reduzido o *dataset.* Os resultados obtidos sob estas condições estão descritos a seguir.

Utilizando o *dataset* do MAFAULDA completo, com 3330 dados de desbalanceamento e 490 dados para condições normais, a acurácia obtida foi de 76,81%, resultado considerado insatisfatório. Visando otimizar o algoritmo, buscamos alternativas que resultassem em uma classificação mais precisa dos experimentos.

Foi observado que o número de dados de desbalanceamento fornecidos à máquina para treinamento era muito superior àquele para condições normais. Como teste, decidimos utilizar quantidade reduzida de dados de desbalanceamento, 1500 ao invés dos 3330 iniciais, e manter os 490 dados normais. A acurácia obtida com estas configurações foi de 79,71%. Considerando que houve melhora na precisão da classificação, alteramos o *dataset* de forma a fornecer a mesma quantidade de dados para desbalanceamento e condições normais, 490 dados para cada status. Neste caso, a acurácia chegou a 85,51%.

Apesar de neste momento termos obtido um resultado considerado satisfatório, testamos uma última alteração para verificar a possibilidade de melhora na precisão. Esta mudança se baseou na generalização do sistema novamente, em que se parte do princípio de que reduzir os dados fornecidos à máquina para treinamento a obrigará a encontrar relações mais fortes entre os dados. Estas seriam, por sua vez, relações gerais de máquinas rotativas e menos específicas ao *setup* experimental utilizado na obtenção dos dados.

De fato, reduzindo os dados de desbalanceamento e de condições normais para 50 e 150, respectivamente, obteu-se uma acurácia de 91,30%, comprovando a hipótese de generalização novamente. O código utilizado na obtenção dos resultados citados anteriormente, dispostos na Tabela 21, está disponível no Apêndice 8.

TABELA 21 - Resultados Finais

Dados Desbalanceamento/Normal	**Acurácia**
3330/490	76,81%
1500/490	79,71%
490/490	85,51%
50/150	91,30%

FONTE: Os Autores, 2021

Das 70 sequências de dados obtidas através de experimentos em laboratório, uma delas apresentou problema no cálculo da rotação, que não condizia com a configurada, e foi excluída do *dataset* a ser classificado. Trata-se do Teste 1 do Mancal 2 para um desbalanceamento com 1 grama e rotação de 2800 RPM. Foram classificados então, 69 casos diferentes, dentre desbalanceamentos com 1 e 4 gramas e condições normais.

A Tabela 22 a seguir apresenta a configuração dos dados que foram classificados de forma errônea.

TABELA 22 - Erros de classificação

Teste	**Mancal**	**Rotação**	**Status**	**Classificação**
1	1	2200 RPM	Desbalanceamento 1g	Normal
2	1	2200 RPM	Desbalanceamento 1g	Normal
1	2	2800 RPM	Normal	Desbalanceamento
2	2	2800 RPM	Normal	Desbalanceamento
1	1	1600 RPM	Desbalanceamento 1g	Normal
2	1	1600 RPM	Desbalanceamento 1g	Normal

FONTE: Os Autores, 2021

Nota-se que os casos 1 e 2, 3 e 4 e 5 e 6 ocorreram para as mesmas configurações, divergindo apenas por se tratarem de testes distintos. Essa constatação levanta a hipótese de problema na aferição dos dados para estes casos em particular e não uma imprecisão do classificador, apesar dos gráficos e dos valores das *features* não apresentarem discrepâncias substanciais.

O resultado final deste trabalho consiste em um classificador final, disponibilizado no Apêndice 9, que permite ao usuário selecionar arquivos diretamente de seu computador para classificá-los. O formato do arquivo deve ser compatível ao utilizado por este trabalho, que foi gerado de forma automática pelo software de aferição citado na seção de metodologia.

5 CONCLUSÕES E PRÓXIMOS PASSOS

Observou-se através dos resultados dispostos na seção 4.1 que, apesar de amplamente disponibilizados, bancos de dados podem conter imprecisões que podem acarretar em resultados falsos. Fazem-se fundamentais, portanto, as etapas de tratamento e análise crítica dos dados a serem utilizados.

A seção 4.2 permite concluir que é possível alcançar alta precisão na classificação de falhas em máquinas rotativas por intermédio de modelos de aprendizado de máquina, uma vez que seja possível coletar dados experimentais do sistema mecânico. Isso mostra que as *features* extraídas, juntamente com o algoritmo, são capazes de identificar e replicar padrões de vibração para diversas configurações da bancada.

Há, entretanto, um ponto negativo quando levada em consideração a aplicabilidade deste método na indústria. Isso porque não é viável interromper o funcionamento de uma máquina para realizar ensaios que sirvam como *dataset* base para classificação posterior. Além disso, considera-se o fato que a alta precisão obtida para o MAFAULDA limita-se a um setup idêntico ao apresentado por Marins et. al, 2018, não sendo possível aplicá-lo para outras máquinas com a mesma acurácia. Isso ocorre quando uma quantidade significativa de *features* é utilizada, visto que se atinge alta especificidade e limita-se sua aplicabilidade, como apresentado na seção de resultados e discussões. Contornar este problema é possível através da realização de etapas de análise e tratamento dos dados, bem como generalização desenvolvida neste trabalho.

Identifica-se, portanto, que este trabalho tem grande importância como referência para a aplicação da manutenção preditiva na indústria, visto que disponibiliza um classificador final generalizado, que passou previamente por treinamento e coleta de dados.

Seguindo esta linha de pensamento e levando em consideração os resultados da seção 4.6, provou-se possível realizar a classificação do conjunto de dados obtidos experimentalmente. Isso apesar de apresentar perdas na precisão com ganhos em generalização e aplicabilidade.

5.1 CONSIDERAÇÕES FINAIS

Os resultados dispostos na seção de resultados e discussões concluem o objetivo geral e os objetivos específicos propostos para o desenvolvimento deste trabalho. Devido a restrições relacionadas ao setup experimental, o qual não permitiu a simulação das condições de operação do MAFAULDA, como desalinhamento vertical e horizontal e falhas de rolamento, não foi possível concluir por completo o objetivo específico “coletar dados reais através de experimentos controlados em bancadas para verificar a acurácia do algoritmo”.

De qualquer forma, consideram-se satisfatórios os resultados obtidos para este objetivo, visto que foi possível classificar os experimentos realizados em laboratório para as condições de desbalanceamento e normal, além de validar o algoritmo para erros de desalinhamento e falhas de rolamento através do método de *cross validation*. Espera-se, desta forma, uma boa resposta em dados experimentais após análise e tratamento dos dados pertinentes.

5.2 PRÓXIMOS PASSOS

Realizar experimentos em bancadas que possibilitem a simulação das falhas de desalinhamento vertical e horizontal e de rolamento, e verificar se a inclusão destes dados interfere na precisão da classificação.

Expandir rol de experimentos a fim de classificar novas máquinas utilizando como *dataset* de treinamento a junção dos dados obtidos experimentalmente, tanto os já obtidos como aqueles a serem coletados, com os dados disponibilizados pelo MAFAULDA.

Buscar, através dos novos experimentos realizados e citados anteriormente, incluir o eixo Z na análise das falhas, visto que este foi desconsiderado no classificador final.

Utilizar este trabalho como referência para aplicação prática dos resultados e conclusões na manutenção preditiva de máquinas rotativas na indústria.

Aplicar os conceitos da Indústria 4.0, empregando acelerômetros e outros sensores para monitorar o maquinário e coletar seus dados de vibração, possibilitando identificar falhas antes de ocorrerem.

REFERÊNCIAS

AYVAZ, Serkan; ALPAY, Koray. Predictive maintenance system for production lines in manufacturing: A Machine Learning approach using IoT data in real-time. Expert Systems with Applications, 2021, 173. Jg., S. 114598.

BAHRIN, Mohd Aiman Kamarul et al. Industry 4.0: A review on industrial automation and robotic. Jurnal Teknologi, v. 78, n. 6-13, 2016.

BEZERRA, Roberto de Araujo et al. Detecção de falhas em rolamentos por análise de vibração. 2004.

BJØRNØ, Leif. Applied underwater acoustics. Amsterdam: Elsevier, 2017.

BOUGRAIN, Laurent. Practical introduction to artificial neural networks. IFAC Proceedings Volumes, 2004, 37. Jg., Nr. 15, S. 347-352.

BUMBLAUSKAS, Daniel et al. Smart Maintenance Decision Support Systems (SMDSS) based on corporate big data analytics. Expert systems with applications, v. 90, p. 303-317, 2017.

CACHADA, Ana et al. Maintenance 4.0: Intelligent and predictive maintenance system architecture. In: 2018 IEEE 23rd international conference on emerging technologies and factory automation (ETFA). IEEE, 2018. p. 139-146.

DALZOCHIO, Jovani, et al. Machine Learning and reasoning for predictive maintenance in Industry 4.0: Current status and challenges. Computers in Industry, 2020, 123. Jg., S. 103298.

DAVENPORT, Thomas H. Analytics 3.0. Harvard business review, v. 91, n. 12, p. 64-72, 2013.

GANDHI, Rohith Support Vector Machine — Introduction to Machine Learning Algorithms, 2018 - disponivel em: < https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 >

GOYAL, D.; PABLA, B. S. The vibration monitoring methods and signal processing techniques for structural health monitoring: a review. Archives of Computational Methods in Engineering, v. 23, n. 4, p. 585-594, 2016.

KIRAL, Zeki; KARAGÜLLE, Hira. Simulation and analysis of vibration signals generated by rolling element bearing with defects. Tribology International, v. 36, n. 9, p. 667-678, 2003.

KOOHANG, Alex; NORD, Jeretta Horn. Critical components of data analytics in organizations: A research model. Expert Systems with Applications, v. 166, p. 114118, 2021.

KRÖSE, Ben, et al. An introduction to neural networks. 1993.

LASI, Heiner et al. Industry 4.0. Business & Information systems engineering, v. 6, n. 4, p. 239-242, 2014.

LEE, Jay; KAO, Hung-An; YANG, Shanhu. Service innovation and smart analytics for industry 4.0 and big data environment. Procedia Cirp, v. 16, p. 3-8, 2014.

LEES, Arthur W. Vibration Problems in Machines: Diagnosis and Resolution. CRC Press, 2016.

MAIS, Jason. Spectrum analysis: the key features of analyzing spectra. SKF USA, Inc, 2002.

MANDOT, Pushkar What is LightGBM, How to implement it? How to fine tune the parameters? , 2017 - disponível em: <https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc> Acesso em: 07 de Dezembro de 2021.

MANYIKA, James, et al. Big data: The next frontier for innovation, competition, and productivity. McKinsey Global Institute, 2011.

MARINS, Matheus A. et al. Improved similarity-based modeling for the classification of rotating-machine failures. Journal of the Franklin Institute, v. 355, n. 4, p. 1913-1930, 2018.

MATHIAS, M. H. Ferramentas de diagnóstico de máquinas, Guaratinguetá, 20--.

Disponivel em: <https://slidetodoc.com/ambiente-multimdia-de-suporte-disciplina-de-psgraduao-ferramentas-4/>. Acesso em: 19 de Agosto de 2021.

MOBLEY, R. Keith. An introduction to predictive maintenance. Elsevier, 2002.

NAVALANI, Support Vector Machine Classification in Scikit-learn, 2020 - disponível em:

<https://avinashnavlani.medium.com/support-vector-machine-classification-in-scikit-learn-3800bc4979ce> Acesso em: 07 de Dezembro de 2021.

PATURI, Uma Maheshwera Reddy; CHERUKU, Suryapavan. Application and performance of machine learning techniques in manufacturing sector from the past two decades: a review. Materials Today: Proceedings, 2020.

PENEDO, Antonio Sérgio Torres. Estrutura de rede neural. Disponível em: <https://www.researchgate.net/figure/Figura-1-Estrutura-da-rede-neural-Os-3-neuronios-da-camada-de-saida-formam-a-matriz-3x1_fig1_324555121>. Acesso em: 07 dez. 2021.

QIN, S. Joe; CHIANG, Leo H. Advances and opportunities in machine learning for process data analytics. Computers & Chemical Engineering, 2019, 126. Jg., S. 465-473.

RANDALL, Robert Bond. Vibration-based condition monitoring. John Wiley & Sons, 2011.

SELCUK, Sule. Predictive maintenance, its implementation and latest trends. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, v. 231, n. 9, p. 1670-1679, 2017.

SINGH, Jaswinder; DHIMAN, Gaurav. A survey on machine-learning approaches: Theory and their concepts. Materials Today: Proceedings, 2021.

STEURTEWAGEN, Bram; VAN DEN POEL, Dirk. Adding Interpretability to Predictive Maintenance by Machine learning on Sensor Data. Computers & Chemical Engineering, 2021, S. 107381.

TANDON, Naresh; CHOUDHURY, Achintya. A review of vibration and acoustic measurement methods for the detection of defects in rolling element bearings. Tribology international, v. 32, n. 8, p. 469-480, 1999.

VADAPALLI, Pavan. Machine learning vs Data Analytics: Difference Between machine learning and Data Analytics. Disponível em: https://www.upgrad.com/blog/machine-learning-vs-data-analytics/. Acesso em: 18 ago. 2021.

VAIDYA, Saurabh; AMBAD, Prashant; BHOSLE, Santosh. Industry 4.0–a glimpse. Procedia manufacturing, v. 20, p. 233-238, 2018.

YIU, Tony Understanding Random Forest, 2019 - disponível em: <https://towardsdatascience.com/understanding-random-forest-58381e0602d2> Acesso em: 07 de Dezembro de 2021.

APÊNDICE 1 – CÓDIGO DE FUNÇÕES BASE

Este algoritmo contém as funções que servem como suporte para os algoritmos que seguirão.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy import fft
import plotly.graph_objects as go
from scipy import signal
import os


fs = 50000 #[Hz] frequência de amostragem
T_amost =5 #[s]  tempo total de amostragem
Ts=1/fs    #[s]  Período de amostragem
t = np.arange(0,T_amost,Ts) # vetor tempo de 0 a 5s


# le o arquivo .csv do banco de dados MAFAULDA e aplica um fil
tro  passa faixa [10 - 5500] Hz nas colunas de aceleração
def read_file(file):
  colunas = ['tacometro', 'axial_1','radial_1', 'tangencial_1'
,'axial_2','radial_2', 'tangencial_2','microfone']
  csv = pd.read_csv(file, header=None, names=colunas)
  sos = signal.butter(N=10, Wn=[10,5500], btype='bandpass', fs
=fs, output='sos')
  for col in csv.columns[1:-1]:
    csv[col] = signal.sosfilt(sos, csv[col])
  return csv

#filtra o sinal do tacômetro para valores de 0 ou 5
def filtro(sinal, low=3, high=3):
  sinal[sinal<low]=0
  sinal[sinal>high]=5
  return sinal

#Retorna a frequência de rotação do sinal do tacômetro analógi
co
def get_rotation(sinal, n=10):
  picos, count, temp, start = 0, 0, 5, False
  sinal= filtro(sinal)
  for value in sinal:
    if (temp==0) and (value== 5.0):
      start=True
      picos+=1
```

```
      if picos > n: break
    temp = value
    if start: count+=1
  return  n/(count*Ts)

#Retorna a frequencia de rotação do sinal do tacometro analógi
co   com auxilio da função find_peaks
def find_rotation(sinal, h=0.5):
  return signal.find_peaks(to_fft(sinal.values),height=h)[0][0
]/ (sinal.shape[0]/fs)

# Retorna a fft
def to_fft(y):
  return np.abs(fft.fft(y))[0:y.shape[0]//2] / (y.shape[0]//2)

#plota os gráficos de FFTs para as colunas selecionadas
def plot_fft(df:pd.DataFrame, plot_cols:list, title=None):
  fig = go.Figure()
  for col in plot_cols:
    y_fft= to_fft(df[col].values)
    fig.add_trace(go.Scatter(
        x=fft.fftfreq(df.shape[0],Ts),
        y= y_fft,
        name = col
    ))
  if title !=None:
    fig.update_layout(title=title)
  fig.show()

#--------------Estatísticas no domínio do tempo---------------
def pico(x):
  return np.max(abs(x))
def RMS(x):
  return np.sqrt(np.mean(x**2))
def crista(x):
  return pico(x)/RMS(x)
def curtose(x):
  a = 1/x.shape[0] * np.sum((x-np.mean(x))**4)
  b = (1/x.shape[0] * np.sum((x-np.mean(x))**2))**2
  return a/b
#----------------------------------------------------------

#
Retorna as ampiltudes de vibração para as frequencias selecion
adas
def get_peaks(fft, freqs, rot, delta=3):
  values = []
  xvar = fft.shape[0]//(fs//2)
  ref = (np.array(freqs)*rot*xvar).astype(int)
```

```
  for freq in ref:
    values.append(np.max(fft[freq-delta:freq+delta+1]))
  return values

# Divide o dataframe em N partes------------------------------
-----
def split_df(df, n):
  df_splited = []
  y= df.shape[0]//n
  for x in range(n):
    df_splited.append(df.iloc[x*y:(x+1)*y])
  return df_splited

# Retorna o dataframe do MAFAULDA compilado sem levar em conta
 a posição do acelerometro
def split_df_2(df, n=5):
  df_splited = []
  for sample in split_df(df, n):
    df_splited.append(sample.iloc[:,1:4])
    df_splited.append(sample.iloc[:,4:7])

  return df_splited
```

APÊNDICE 2 – CÓDIGO PARA EXTRAÇÃO DE *FEATURES* MAFAULDA

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
PATH = '/content/drive/MyDrive/MAFAULDA/'


os.listdir(PATH)


for file in os.listdir(PATH):
  print(file)
  for folder in os.listdir(PATH+file):
    print(os.listdir(f'{PATH}{file}/{folder}'))



features = []
freqs = [1,2,3,4,5,6] #lista de picos a serem extraidos da FFT
i =0

for file in os.listdir(PATH):
  for folder in os.listdir(PATH+file):
    for csv in os.listdir(f'{PATH}{file}/{folder}'): #para cad
a arquivo do banco de dados
      df = read_file(f'{PATH}{file}/{folder}/{csv}')
      rot= find_rotation(df.tacometro)
      for sample in split_df(df,5):
        features.append([])
        features[i].append(csv) #adiciona o nome
        features[i].append(rot) #adiciona a rotação
        for col in ['axial_1','radial_1', 'tangencial_1','axia
l_2','radial_2', 'tangencial_2']: # para cada coluna
          features[i].append(pico(sample[col])) #adiciona o pi
co
          features[i].append(RMS(sample[col]))  #adiciona o RM
S
          features[i].append(crista(sample[col])) #adiciona o
fator de Crista
          features[i].append(curtose(sample[col])) #adiciona a
 curtose
          for peak in get_peaks(to_fft(sample[col].values),fre
qs,rot): #para cada pico selecionado
            features[i].append(peak) #adiciona o pico
        features[i].append(file) #adicina o status da maquina
[normal, desbalanceado, vertical, horizontal]
        i +=1
df_features = pd.DataFrame(features)
print(df_features.shape)
```

```
df_features.head()


# criando a lista de colunas e simplificando as direçoes em X,
 Y e Z
acel = ['X1', 'Z1', 'Y1','X2', 'Z2', 'Y2']
# ['axial_1','radial_1', 'tangencial_1','axial_2','radial_2',
'tangencial_2']
colunas=[]
for col in acel:
  colunas.append(f'{col}_peak')
  colunas.append(f'{col}_RMS')
  colunas.append(f'{col}_crista')
  colunas.append(f'{col}_curtose')
  for freq in range(1,7):
    colunas.append(f'{col}_{freq}x')
colunas = ['name','rotation'] + colunas + ['status']

df_features.columns= colunas
df_features.to_csv('/content/drive/My Drive/TCC/CSVs/features_
sampled_5_v4.csv', index=False)
```

APÊNDICE 3 – CÓDIGO PARA TREINAMENTO DE APRENDIZADO DE MÁQUINA

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split, GridSear
chCV
from sklearn.svm import SVC
from sklearn.pipeline import make_pipeline
from sklearn.metrics import accuracy_score, f1_score
from sklearn.compose import make_column_transformer
from sklearn.ensemble import RandomForestClassifier
from lightgbm import LGBMClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier


#Carregando arquivos do google drive
from google.colab import drive
drive.mount('/content/drive')

FILE = '/content/drive/My Drive/UFPR/TCC/CSVs/compiled_3.csv'
raw = pd.read_csv(FILE)

df1 = raw.copy()
TARGET = 'status_comb'
# excluindo a coluna de status das features
features = df1.drop([TARGET], axis=1).columns
#excluindo a coluna com os picos
features = [col for col in features if 'peak' not in col]
#selecionando as colunas com as features do tempo
time_cols = [col for col in features if ('RMS' in col) or ('cr
ista' in col) or ('curtose' in col) or (col=='rotation')]
#selecionando as colunas com as features da FFT
fft_cols = [col for col in features if col not in time_cols]

# separando os dados em features e target para treino e teste
de 30%
df1= raw.sample(frac=1)
X = df1[features]
y = df1[TARGET]
X_train, X_test, y_train, y_test = train_test_split(X, y, test
_size=0.3, random_state=42)
scaler = StandardScaler()
```

```
# 1° modelo SVM
clf1 = make_pipeline(scaler,SVC())
clf1.fit(X_train,y_train)
y_pred= clf1.predict(X_test)
print(accuracy_score(y_test,y_pred))

# Parametors para o hyperparamter tuning
params = {
    'svc__C':range(1,10),
    'svc__degree': range(2,6)
}
grid1 = GridSearchCV(clf1,params, scoring='accuracy', n_jobs=-
1, cv=5, verbose=1)
grid1.fit(X,y)
print(grid1.best_params_, grid1.best_score_)

# 2° Modelo LGBM
clf2 = make_pipeline(scaler, LGBMClassifier())
clf2.fit(X_train,y_train)
y_pred= clf2.predict(X_test)
print(accuracy_score(y_test,y_pred))

# Parametros para o hyperparamter tuning
params = {
    'lgbmclassifier__num_leaves' : [7,20,30,40,50,60],
    'lgbmclassifier__max_depth': [-1,2,3,4,5,6]
}
grid2 = GridSearchCV(clf2,params, scoring='accuracy', n_jobs=-
1, cv=5, verbose=1)
grid2.fit(X,y)
print(grid2.best_params_, grid2.best_score_)

# Analise das features mais importantes para o LGBM
feature_imp = pd.DataFrame(zip(grid2.best_estimator_[1].featur
e_importances_,X.columns), columns=['Value','Feature'])
plt.figure(figsize=(20, 10))
sns.barplot(x="Value", y="Feature", data=feature_imp.sort_valu
es(by="Value", ascending=False), palette='YlGnBu_r')
plt.title('LGBM Features importance')
plt.tight_layout()
plt.show()

#compilado das features importance para as direções XYZ
feature_imp['dir']= feature_imp.Feature.str[:2]
dir_importance=feature_imp.groupby('dir').sum().drop('ro').sor
t_values('Value', ascending=False).reset_index()
plt.figure(figsize=(5,2))
plt.title('Feature Importance compiled LGBM')
```

```
sns.barplot(x='Value', y='dir', data=dir_importance, palette='
YlGnBu_r')
plt.show()
#Criando um scaler personalizado
min_max_scaler = MinMaxScaler((0,int(df1[fft_cols].max().max()
)))
transformer = make_column_transformer(
    (min_max_scaler, time_cols),
    remainder='passthrough'
)
clf3 = make_pipeline(transformer, LGBMClassifier())
clf3.fit(X_train,y_train)
y_pred= clf3.predict(X_test)
print(accuracy_score(y_test,y_pred))

#Parametros para o hyperparamter tunning
params = {
    'lgbmclassifier__num_leaves' : [7,20,30,40,50,60],
    'lgbmclassifier__max_depth': [-1,1,2,3,4,5,6]
}
grid3 = GridSearchCV(clf3,params, scoring='accuracy', n_jobs=-
1, cv=5, verbose=1)
grid3.fit(X,y)
print(grid3.best_params_, grid3.best_score_)

#3° modelo Random Forest
clf4 = make_pipeline(transformer,RandomForestClassifier())
clf4.fit(X_train,y_train)
y_pred= clf4.predict(X_test)
print(accuracy_score(y_test,y_pred))
#Parametros para o hyperparamter tuning

params = {
    'randomforestclassifier__bootstrap': [True, False],
    'randomforestclassifier__max_depth': [5,10,20,30],
    'randomforestclassifier__max_features': ['auto','sqrt'],
    'randomforestclassifier__min_samples_leaf': [1,2,4],
    'randomforestclassifier__min_samples_split': [2,5,7,10],
    'randomforestclassifier__n_estimators': [50,100,200,300,40
0,500]
}
grid4= RandomizedSearchCV(clf4,params,n_iter = 500,scoring='ac
curacy',n_jobs=-1, cv=5, verbose=1)
grid4.fit(X,y)
print(grid4.best_params_, grid4.best_score_)
```

APÊNDICE 4 – CÓDIGO PARA *MLP TRAIN TEST SPLIT*

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from google.colab import drive
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV

#Carregando arquivo de features do drive
drive.mount('/content/drive')
FILE = '/content/drive/MyDrive/UFPR/TCC/CSVs/compiled_3.csv'

#separando as features e excluindo o status
TARGET = 'status_comb'
features = df1.drop(TARGET, axis=1).columns
features = [col for col in features if 'peak' not in col]

X= df1[features]
y=pd.get_dummies(df1[TARGET]) # Realizando o One Hot Encodding
input_shape = X.shape[1]
output_shape = y.shape[1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test
_size=0.3, random_state=42)

#realizando o escalonamento dos dados
scaler = StandardScaler()
X_train= scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Criando um modelo Keras
model= keras.Sequential([
                         layers.Dense(100, activation='relu',
input_shape=[input_shape]),
                         layers.Dropout(0.5),
                         layers.Dense(50, activation='relu'),
                         layers.Dense(50, activation='relu'),
                         layers.Dense(output_shape, activation
='softmax')
])
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics= ['accuracy'])
#Criando Early Stopping
```

```
es = keras.callbacks.EarlyStopping(
    patience = 40,
    min_delta = 0.0001,
    restore_best_weights = True
)
# treinamento MLP
history = model.fit(
    X_train, y_train,
    validation_data=(X_test, y_test),
    batch_size=32,
    epochs=200,
    verbose = 1,
    callbacks = es
)
results = pd.DataFrame(history.history)
results.loc[10:,['accuracy', 'val_accuracy']].plot()
```

APÊNDICE 5 – CÓDIGO PARA O *HYPERPARAMETER TUNING MLP*

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from google.colab import drive
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSear
chCV
from sklearn.model_selection import GridSearchCV
from scikeras.wrappers import KerasClassifier
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import accuracy_score, f1_score

# Carregando arquivo de features do drive
drive.mount('/content/drive')
FILE = '/content/drive/MyDrive/UFPR/TCC/CSVs/compiled_3.csv'

raw= pd.read_csv(FILE)
df1 = raw.sample(frac = 1, random_state=42)

# Separando as features e excluindo o status e os picos
TARGET = 'status_comb'
features = df1.drop(TARGET, axis=1).columns
features = [col for col in features if 'peak' not in col]

X= df1[features]
y=pd.get_dummies(df1[TARGET]) # Realizando o One Hot Encoding

input_shape = X.shape[1]
output_shape = y.shape[1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test
_size=0.3, random_state=42)
input_shape,output_shape

# Escalonamento dos dados
scaler = StandardScaler()
X_train= scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Função para criação do modelo Keras
def create_model(drop, n1,n2,n3, activation):
  model= keras.Sequential([
```

```
                            layers.Dense(n1, activation=activatio
n, input_shape=[input_shape]),
                            layers.Dense(n2, activation=activatio
n),
                            layers.Dropout(drop),
                            layers.Dense(n3, activation=activatio
n),
                            layers.Dense(output_shape, activation
='softmax')
  ])

  return model

# Hyperparameter tuning
es = keras.callbacks.EarlyStopping(
    monitor='loss',
    patience = 10,
    restore_best_weights = True
)
model = KerasClassifier(model=create_model, callbacks=[es], dr
op=None,n1=None,n2=None,n3=None, activation=None, epochs=200,
verbose=0)
params={
    'kerasclassifier__batch_size':[16,32,64,128],
    'kerasclassifier__loss':['categorical_crossentropy','poiss
on','categorical_hinge'],
    'kerasclassifier__optimizer':['SGD', 'Adam'],
    'kerasclassifier__drop':[0.1,0.3,0.5,0.7,0.9],
    'kerasclassifier__n1':[100,50,30],
    'kerasclassifier__n2':[100,50,30],
    'kerasclassifier__n3':[100,50,30],
    'kerasclassifier__activation':['softmax', 'relu', 'sigmoid
', 'tanh'],

    }
clf = make_pipeline(scaler,model)
grid = RandomizedSearchCV(clf,params,n_iter = 100,scoring='acc
uracy',n_jobs=-1, cv=5, verbose=1)
grid.fit(X,y)

# Resultado dos tuning
print(grid.best_params_, grid.best_score_)
```

APÊNDICE 6 – CÓDIGO PARA GERAR GRÁFICO DE FFT E TEMPO

```
import pandas as pd
import numpy as np
import os
import plotly.graph_objects as go
from scipy import signal, fft

# informações da coluna do excel
start =84
endfreq = 3284
end =8275
fs = 1024

# Definindo funções
def read_file(name):
    excel = pd.read_excel(name, header=None)
    ffts = excel.iloc[start:endfreq, [1,2,3,6,7,10,11]]
    ffts['FFT_X']=abs(ffts[2] + ffts[3]*1j)
    ffts['FFT_Y']=abs(ffts[6] + ffts[7]*1j)
    ffts['FFT_Z']=abs(ffts[10] + ffts[11]*1j)
    ffts = ffts.drop([2,3,6,7,10,11], axis=1).set_index(1)
    ffts.iloc[:80]=0

    sos = signal.butter(N=10, Wn=10, btype='highpass', fs=fs,
output='sos')
    time = excel.iloc[start:end, [13,14,17,20]].rename(columns
={14:'X',17:'Y',20:'Z'})
    for col in time.columns[1:]:
        time[col]= signal.sosfilt(sos, time[col])
        time[col]= time[col]

    return ffts, time.set_index(13)

def read_sem_filtro(name):
    excel = pd.read_excel(name, header=None)
    time = excel.iloc[start:end, [13,14,17,20]].rename(columns
={14:'X',17:'Y',20:'Z'})

    return time.set_index(13)

# Carregando arquivo de features do drive
from google.colab import drive
drive.mount('/content/drive')

PATH = '/content/drive/MyDrive/UFPR/TCC/Python/Experimentos/29
-11/'
```

```
# Plotando a FFT
fig = go.Figure()
FILE = PATH + 'Normal/Normal/2800 RPM - Teste 1 - Mancal 1.xls
x'
_,df =read_file(FILE)

col = 'X'
fig.add_trace(go.Scatter(
    x=fft.fftfreq(df.shape[0],1/fs),
    y=to_fft(df[col].values),
))

fig.update_layout(title= 'FFT - 3400RPM - Mancal 2')
fig.show()

fig2= go.Figure()
_,df =read_file(FILE)
df_sf = read_sem_filtro(FILE)

col = 'Y'
fig2.add_trace(go.Scatter(
    x=df_sf.index,
    y=df_sf[col],
    name = 'Sem Filtro'
))
fig2.add_trace(go.Scatter(
    x=df.index,
    y=df[col],
    name='Com Filtro'
))

fig2.update_layout(title= 'Aceleração x Tempo - 3400RPM - Manc
al 2')
fig2.show()
```

APÊNDICE 7 – CÓDIGO PARA EXTRAÇÃO DE *FEATURES* DOS EXPERIMENTOS

```
import pandas as pd
import numpy as np
import os
import plotly.graph_objects as go
from scipy import signal, fft
fs = 1024

# Definindo função para extrair dados da planilha de Excel
def read_file(name):
  start =84
  end =8275
  fs = 1024
  excel = pd.read_excel(name, header=None)
  sos = signal.butter(N=10, Wn=10, btype='highpass', fs=fs, ou
tput='sos')
  time = excel.iloc[start:end, [13,14,17,20]].rename(columns={
14:'X',17:'Y',20:'Z'})
  for col in time.columns[1:]:
      time[col]= signal.sosfilt(sos, time[col])
  return time.set_index(13)

#Carregando arquivo de features do drive
from google.colab import drive
drive.mount('/content/drive')

PATH = 'content/drive/MyDrive/UFPR/TCC/Python/Experimentos/29-
11/'

# Extração das features
features = []
i=0
freqs = [1,2]

for status in os.listdir(PATH):
  for folder in os.listdir(f'{PATH}{status}'):
    for file in os.listdir(f'{PATH}{status}/{folder}'):
      features.append([])
      time = read_file(f'{PATH}{status}/{folder}/{file}')
      rot=find_rotation(time['X'],20e-9)
      features[i].append(rot)
      for col in ['X', 'Y']:
        features[i].append(crista(time[col]))
        features[i].append(curtose(time[col]))
```

```
        for peak in get_peaks(to_fft(time[col].values),freqs,r
ot,2):
          RMS1=RMS(time[col])
          features[i].append(peak/RMS1)
      features[i].append(status)
      i+=1


acel = ['X', 'Y']
colunas=[]
for col in acel:
  colunas.append(f'{col}_crista')
  colunas.append(f'{col}_curtose')
  for freq in [1,2]:#freqs:
    colunas.append(f'{col}_{freq}x')
colunas = ['rotation'] + colunas + ['status']
print(colunas)

df_exp = pd.DataFrame(features, columns=colunas)
# Excluindo arquivo com erro
df_exp.drop(13,inplace=True)

# Salvando o arquivo no drive
name = 'experimentos29-11_v2.csv'
df_exp.to_csv('/content/drive/MyDrive/UFPR/TCC/CSVs/' + name ,
 index=False)
```

APÊNDICE 8 – CÓDIGO PARA CLASSIFICAÇÃO DOS EXPERIMENTOS

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
import pickle
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split, GridSear
chCV
from sklearn.svm import SVC
from sklearn.pipeline import make_pipeline
from sklearn.metrics import accuracy_score, f1_score
from sklearn.compose import make_column_transformer
from sklearn.ensemble import RandomForestClassifier
from lightgbm import LGBMClassifier
from sklearn.neural_network import MLPClassifier
from google.colab import drive

# Definindo os diretórios
drive.mount('/content/drive', force_remount=True)
PATH = '/content/drive/MyDrive/UFPR/TCC/CSVs/'
FILE = PATH+'features_sampled_5_split_2.csv'
EXPERIMENTO = PATH+ 'features_experimento_1.csv'
EXPERIMENTO2 = PATH+ 'experimentos29-11_v2.csv'

# Carregando o arquivo
df_experimento = pd.read_csv(EXPERIMENTO)
df1= pd.read_csv(FILE).sample(frac=1, random_state=42)
df_experimento2 =  pd.read_csv(EXPERIMENTO2)
print(df_experimento.shape[0],df_experimento2.shape[0])

#Separando os dados normais e desbalanceados do MAFAULDA
imbalance = df1[(df1.status=='imbalance/')][0:50]
normal = df1[(df1.status=='normal/')] [0:50]
df1 = pd.concat([imbalance,normal])
df1 = df1.sample(frac = 1, random_state=42).reset_index()

# Separando as colunas para seren normalizadas (col/RMS)

normalize_cols = [col for col in df1.columns if col[-
2:]=='1x' or col[-2:]=='2x']
X_cols = [col for col in normalize_cols if col[0] =='X']
Y_cols=[col for col in normalize_cols if col[0] =='Y']


# normalizando as colunas do MAFAULDA
```

```
for i in range(df1.shape[0]):
  df1.loc[i,X_cols] = df1.loc[i,X_cols] / df1.X_RMS.loc[i]
  df1.loc[i,Y_cols] = df1.loc[i,Y_cols] / df1.Y_RMS.loc[i]

# normalizando as colunas dos experimentos
for i in range(df_experimento.shape[0]):
  df_experimento.loc[i,X_cols] = df_experimento.loc[i,X_cols]
/ df_experimento.X_RMS.loc[i]
  df_experimento.loc[i,Y_cols] = df_experimento.loc[i,Y_cols]
/ df_experimento.Y_RMS.loc[i]


# Definindo as features e Target
TARGET = 'status'
features = ['rotation','X_crista', 'X_1x', 'X_2x','Y_crista','
Y_1x', 'Y_2x', 'X_curtose', 'Y_curtose']

X= df1[features]
y=df1[TARGET].apply(lambda x: 1 if x[:3]=='imb' else 0)
#-----------------------------------------------------------
experimentos_all = pd.concat([df_experimento[features + [TARGE
T]],df_experimento2[features + [TARGET]] ])
X_predict = experimentos_all[features]
y_true = experimentos_all[TARGET].apply(lambda x: 1 if x[:3]==
'Des' else 0)

# Colunas que serão escalonadas
scale_cols = [col for col in features if col not in normalize_
cols]
scaler = MinMaxScaler((0,1))
transformer = make_column_transformer(
    (scaler, scale_cols),
    remainder='passthrough',
)
scale_cols

# treinando modelo
my_pipe = make_pipeline(
    transformer,
    RandomForestClassifier(
        bootstrap= False,
        max_depth= 30,
        max_features= 'auto',
        min_samples_leaf= 1,
        min_samples_split= 2,
        n_estimators= 200,
        random_state=42
    )
)
```

```
my_pipe.fit(X,y)
pred = my_pipe.predict(X_predict)
pred

accuracy_score(y_true,pred)

# Gerando lista de nomes na mesmos ordem que os arquivos trein
ados
EXPERIMENTOS = '/content/drive/MyDrive/UFPR/TCC/Python/Experim
entos/'
names = []

for exp in ['25-10/','29-11/']:
  for status in os.listdir(EXPERIMENTOS+exp):
    for folder in os.listdir(f'{EXPERIMENTOS+exp}{status}'):
      for file in os.listdir(f'{EXPERIMENTOS+exp}{status}/{fol
der}'):
        names.append(f'{folder}- {file[:-5]}')
names.remove('1g- 2800 RPM - Teste 1 - Mancal 2')
len(names)

#Criando dataframe com os aqruivos erroneos
df = [names,y_true, pred]
clf_df = pd.DataFrame(df).transpose().rename(columns = {0:'nam
es', 1:'true',2:'pred'})
clf_df[clf_df.true!= clf_df.pred]

#Salvando o modelo treinado no drive
with open(PATH+ 'rfc_experimento.sav', 'wb') as f:
  pickle.dump(my_pipe, f)
```

APÊNDICE 9 – CLASSIFICADOR FINAL

```
import pandas as pd
import numpy as np
import os
import pickle
from scipy import signal, fft
from google.colab import drive, files

fs = 1024


# Carregando modelo treinado
drive.mount('/content/drive')
PATH = '/content/drive/MyDrive/UFPR/TCC/CSVs/'
with open(PATH+ 'rfc_experimento.sav', 'rb') as f:
  model = pickle.load(f)


# Carrega o arquivo excel
def read_file(name):
  start =84
  end =8275
  fs = 1024
  excel = pd.read_excel(name, header=None)
  sos = signal.butter(N=10, Wn=10, btype='highpass', fs=fs, ou
tput='sos')
  time = excel.iloc[start:end, [13,14,17,20]].rename(columns={
14:'X',17:'Y',20:'Z'})
  for col in time.columns[1:]:
      time[col]= signal.sosfilt(sos, time[col])
  return time.set_index(13)

def to_fft(y):
  return np.abs(fft.fft(y))[0:y.shape[0]//2] / (y.shape[0]//2)

def find_rotation(sinal, h=0.05):
  return signal.find_peaks(to_fft(sinal.values),height=h)[0][0
]/ (sinal.shape[0]/fs)

  #--------------Stats from time domain-------------------
def pico(x):
  return np.max(abs(x))
def RMS(x):
  return np.sqrt(np.mean(x**2))
def crista(x):
  return pico(x)/RMS(x)
def curtose(x):
  a = 1/x.shape[0] * np.sum((x-np.mean(x))**4)
```

```
  b = (1/x.shape[0] * np.sum((x-np.mean(x))**2))**2
  return a/b
#-----------------------------------------------------

#Peaks in FFT
def get_peaks(fft, freqs, rot, delta=3):
  values = []
  xvar = (fs//2)/fft.shape[0]
  ref = (np.array(freqs)*rot/xvar).astype(int)
  for freq in ref:
    values.append(np.max(fft[freq-delta:freq+delta+1]))
  return values


#Extraindo features normalizadas de cada arquivo
features = []
i=0
freqs = [1,2]

fnames = list( files.upload() )

print('UPLOAD CONCLUIDO')
for file in fnames:
  features.append([])
  time = read_file(file)
  rot=find_rotation(time['X'],20e-6)
  features[i].append(rot)
  # if 'Mancal 1' in file:
  #   features[i].append(1)
  # else:
  #   features[i].append(3)
  for col in ['X', 'Y']:
    features[i].append(crista(time[col]))
    for peak in get_peaks(to_fft(time[col].values),freqs,rot,2
):
      RMS1=RMS(time[col])
      features[i].append(peak/RMS1)
  i+=1

# Realizando o predict dos arquivos
predict = pd.DataFrame(features, columns=['rotation', 'X_crist
a', 'X_1x', 'X_2x', 'Y_crista', 'Y_1x', 'Y_2x'])
print('-----------------------------------------------------
------------------')
for name, pred in zip(fnames, model.predict(predict)):
  print(f"{name[:-5]}: {['Normal', 'Desbalanceado'][pred]}")
```