

**UNIVERSIDADE REGIONAL DE BLUMENAU – FURB**  
**CENTRO DE CIÊNCIAS TECNOLÓGICAS**  
**ENGENHARIA ELÉTRICA**

**VINICIO SCHMIDT**

**DETECÇÃO DE FALHAS EM EQUIPAMENTOS BASEADO EM ANÁLISE  
SONORA COM O USO DE TÉCNICAS DE APRENDIZADO DE MÁQUINA**

**BLUMENAU**

**2021**



**VINICIO SCHMIDT**

**DETECÇÃO DE FALHAS EM EQUIPAMENTOS BASEADO EM ANÁLISE  
SONORA COM O USO DE TÉCNICAS DE APRENDIZADO DE MÁQUINA**

Trabalho de Conclusão de Curso apresentado ao  
Curso de Engenharia Elétrica da Universidade Re-  
gional de Blumenau como requisito parcial para a  
obtenção do título de Bacharel em Engenharia.

Orientador: Prof. Dr. Luiz Henrique Meyer.

**BLUMENAU**

**2021**

**VINICIO SCHMIDT**

**DETECÇÃO DE FALHAS EM EQUIPAMENTOS BASEADO EM ANÁLISE  
SONORA COM O USO DE TÉCNICAS DE APRENDIZADO DE MÁQUINA**

Trabalho de Conclusão de Curso apresentado ao  
Curso de Engenharia Elétrica da Universidade Re-  
gional de Blumenau como requisito parcial para a  
obtenção do título de Bacharel em Engenharia.

Orientador: Prof. Dr. Luiz Henrique Meyer.

Aprovado em: \_\_\_\_/\_\_\_\_/\_\_\_\_.

---

Presidente: Prof. Luiz Henrique Meyer, Doutor - Orientador, Universidade Regional de Blu-  
menau - FURB

---

Membro: Prof. Marcelo Grafulha Vanti, Doutor, Universidade Regional de Blumenau -  
FURB

---

Membro: Prof. Sandro Geraldo Bagattoli, Mestre, Universidade Regional de Blumenau –  
FURB

Este trabalho é dedicado aos meus familiares, amigos e colegas.

## **AGRADECIMENTOS**

Agradeço aos meus pais, Renato e Izônia, ao meu irmão Matheus, a meus tios Roselene e Luiz e em especial a minha esposa Marli, por todo apoio e estrutura que me proporcionaram durante esta jornada.

Agradeço ainda a todos os professores que tive o prazer de conhecer e que, com diversas contribuições, tornaram o presente trabalho possível.

Os progressos obtidos por meio do ensino são lentos; já os obtidos por meio de exemplos são  
mais imediatos e eficazes.

SENECA, LUCIUS ANNAEUS

## **RESUMO**

Este trabalho apresenta a metodologia para a construção de um protótipo capaz de detectar falhas em equipamentos com o uso de análise sonora por meio de técnicas de ciência de dados e aprendizado de máquina. Além disso, exploram-se técnicas de coleta, processamento, e validação de dados a fim de garantir padrões mínimos de qualidade.

Desse modo, o trabalho é composto por duas grandes etapas. Na primeira etapa, uma introdução ao problema é provisionada bem como uma breve explicação sobre ciência de dados e, em especial, ao aprendizado de máquina. Ainda nesta primeira etapa os principais passos de um projeto de Aprendizado de Máquina é exemplificado.

Já na segunda etapa a construção prática do protótipo é demonstrada. Nesta etapa, o projeto é construído, testado e implantado em produção. Por fim, os resultados são avaliados e explicados.

Palavras-chave: Aprendizado de máquina. Tratamento de dados. Caso real.



## **ABSTRACT**

This work presents the methodology for the construction of a prototype capable of detecting equipment failures using sound analysis through data science and machine learning techniques. In addition, data collection, processing, and validation techniques are explored in order to ensure minimum quality standards.

Thus, the work consists of two major steps. In the first step, an introduction to the problem is provided as well as a brief explanation about data science and, in particular, machine learning. Also in this first step, the main steps of a Machine Learning project are exemplified.

In the second stage, the practical construction of the prototype is demonstrated. At this stage, the project is built, tested and implemented in production. Finally, the results are evaluated and explained.

**Keywords:** Machine learning. Data processing. Real case.

## LISTA DE ILUSTRAÇÕES

Figura 1 – Aprendizado de máquina: um novo paradigma de programação.....	18
Figura 2 – Satisfação de vida vs GDP .....	19
Figura 3 – Regressão linear da Satisfação de vida vs GDP.....	20
Figura 4 – Setup para a captação de dados .....	26
Figura 5 – Defeito de desbalanceamento de eixo simulado .....	27
Figura 6 – DataFrame com os dados carregados .....	30
Figura 7 – Formas de onda para cada classe .....	30
Figura 8 – DataFrame com os dados carregados .....	31
Figura 9 – FFT para as diferentes classes.....	32
Figura 10 – Espectograma .....	33
Figura 11 – Função de transformação dos dados .....	34
Figura 12 – Fluxo de treino e validação dos modelos .....	35
Figura 13 – Matriz de confusão com dados destacados .....	37
Figura 14– Relatório de Classificação.....	39
Figura 15 – Busca em grade .....	40
Figura 16 – Exportação do modelo.....	41
Figura 17 – Importação do modelo.....	41
Figura 18 – Predições com o modelo .....	41
Figura 19 – Interface do sistema em tempo real.....	42
Figura 20 – Interface do sistema em tempo real, para cada defeito analisado .....	43

## **LISTA DE ABREVIATURAS E SIGLAS**

HTML	Linguagem de Marcação de Hipertexto
MP3	Mpeg-layer 3
WAV	Waveform Audio File Format

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO .....</b>	<b>13</b>
1.1	OBJETIVOS.....	15
1.1.1	<b>Objetivos Gerais .....</b>	<b>15</b>
1.1.2	<b>Objetivos Específicos.....</b>	<b>15</b>
1.2	METODOLOGIA .....	16
<b>2</b>	<b>APRENDIZADO DE MÁQUINA.....</b>	<b>17</b>
2.1	O QUE É APRENDIZADO DE MÁQUINA? .....	17
2.1.1	<b>Programação tradicional vs Aprendizagem de máquina .....</b>	<b>17</b>
2.1.2	<b>Como o algoritmo sabe a resposta certa?.....</b>	<b>18</b>
<b>3</b>	<b>FASES DE UM PROJETO DE APRENDIZADO DE MÁQUINA .....</b>	<b>20</b>
3.1	DEFINIÇÃO DO PROBLEMA.....	21
3.2	OBTENÇÃO DOS DADOS .....	22
3.3	EXPLORAÇÃO DOS DADOS .....	23
3.4	PREPARAÇÃO DOS DADOS PARA INCLUIR NO ALGORITMO .....	23
3.5	VALIDAÇÃO DO MELHOR MODELO .....	23
3.6	AJUSTE E COMBINAÇÃO DE MODELOS .....	24
3.7	APRESENTAÇÃO E DOCUMENTAÇÃO DA SOLUÇÃO .....	24
3.8	IMPLANTAÇÃO E MONITORAMENTO DO SISTEMA.....	24
<b>4</b>	<b>PROJETO PRÁTICO.....</b>	<b>26</b>
4.1	COLETA DOS DADOS .....	26
4.1.1	<b>O setup de coleta.....</b>	<b>26</b>
4.1.2	<b>Situações simuladas.....</b>	<b>27</b>
4.1.3	<b>A linguagem utilizada .....</b>	<b>28</b>
4.1.3.1	<i>Quanto dados devem ser gravados?.....</i>	<i>28</i>
4.2	EXPLORAÇÃO DOS DADOS .....	29
4.3	TRANSFORMAÇÃO .....	33
4.4	TESTE E VALIDAÇÃO DO MODELO .....	33
4.4.1	<b>Transformação dos dados.....</b>	<b>34</b>
4.4.1.1	<i>Separação dos conjuntos de treino e teste .....</i>	<i>34</i>
4.4.2	<b>Treino e validação dos algoritmos .....</b>	<b>35</b>
4.4.2.1	<i>Matriz de confusão .....</i>	<i>35</i>
4.4.2.2	<i>Precisão.....</i>	<i>38</i>

4.4.2.3	<i>Recall</i> .....	38
4.4.2.4	<i>F1 Score</i> .....	38
4.4.2.5	<i>Relatório de classificação</i> .....	39
<b>4.4.3</b>	<b>Busca em grade</b> .....	<b>39</b>
4.5	INTEGRAÇÃO COM A FONTE GERADORA DE DADOS .....	40
<b>4.5.1</b>	<b>Exportação do modelo</b> .....	<b>40</b>
<b>4.5.2</b>	<b>Importação do modelo</b> .....	<b>41</b>
<b>4.5.3</b>	<b>Predições com o modelo</b> .....	<b>41</b>
<b>4.5.4</b>	<b>Interface gráfica e exibição dos resultados</b> .....	<b>42</b>
<b>5</b>	<b>CONSIDERAÇÕES FINAIS</b> .....	<b>44</b>
	<b>REFERÊNCIAS</b> .....	<b>45</b>
	<b>APÊNDICE A – Algoritmo de gravação usado para obtenção dos dados</b> .....	<b>48</b>
	<b>APÊNDICE B – Algoritmo para coleta, predição e demonstração dos resultados em tempo real</b> .....	<b>50</b>

## 1 INTRODUÇÃO

“O progresso é a lei da história da humanidade, e o homem está em constante processo de evolução.” (Auguste Comte). Esta frase do filósofo Augusto Comte, apesar de ser usada em um contexto social, é muito ligada ao contexto das revoluções industriais.

Ao observar, por exemplo, a primeira Revolução Industrial pode-se afirmar que foi marcada pela passagem da produção manual para o uso de máquinas com o advento da máquina a vapor. Já com o advento da eletricidade surge a segunda revolução industrial na qual fábricas passaram a ganhar escala e produzir em massa. Não obstante, com o advento das tecnologias da informação, a terceira Revolução Industrial foi marcada pela eletrônica e pelas telecomunicações.

Como observado, o surgimento de tecnologias disruptivas alavanca toda a indústria rumo a uma nova revolução. Atualmente, tecnologias como impressão 3D, internet das coisas, inteligência artificial e nanotecnologias estão fazendo emergir a quarta revolução ou, como é conhecida a indústria 4.0.

Dominar estes conceitos é vital para qualquer empresa que quer existir no futuro, para tanto, o presente trabalho se propõe a demonstrar o uso de algumas destas tecnologias para o ganho de performance no monitoramento e gerenciamento em um dos mais tradicionais elementos das indústrias, o motor elétrico.

“O desenvolvimento do motor elétrico é uma das maiores realizações da indústria de conversão energética moderna [...] no entanto, devido às limitações fundamentais de vida útil do material, deterioração, contaminação, defeitos de fabricação ou danos nas operações, um motor elétrico acabará por entrar em modo de falha” (TOLIYAT, 2017, pg. 11).

Analisando esta frase, é fácil entender a importância de motores elétricos e, ainda mais importante, que não são máquinas indestrutíveis. Devido a sua importância, monitorar, dar manutenção e garantir que os equipamentos funcionem de maneira adequada é um gigante desafio dos setores de manutenção das empresas que afeta diretamente o planejamento de produção e o ciclo de vida de uma empresa.

Conforme descrito em TOLIYAT, 2017, pg. 1-2, as falhas são diversas e podem ocorrer por diversos motivos como falta de limpeza, manutenção inadequada ou simplesmente desgaste por tempo de uso.

Segundo VIB MASTER as principais causas observadas são:

- Transientes de tensão

- Desbalanceamento de fases
- Distorção de harmônicas
- Reflexo dos sinais de saída PWM
- Corrente sigma
- Sobrecarga operacional
- Desalinhamento
- Desbalanceamento de eixos
- Folga do eixo
- Desgaste do rolamento
- Pé suave
- Tensão no eixo

A forma mais comum para evitar estes problemas engloba manutenção preventiva e inspeções como análise de vibração, alinhamento de eixos a laser e balanceamento.

Em TOLİYAT, 2017, pg. 4-5, as estratégias de diagnóstico adotadas na indústria são explicitadas em 4 grandes grupos:

- 1) Diagnóstico de falha baseado em sinal
  - a) Análise de vibração mecânica
  - b) Monitoramento de pulso de choque
  - c) Medição de temperatura
  - d) Análise de ruído acústico
  - e) Monitoramento do campo eletromagnético através da bobina inserida
  - f) Análise de variação de potência de saída instantânea
  - g) Análise infravermelha
  - h) Análise de gás
  - i) Análise de óleo
  - j) Monitoramento de emissão de radiofrequência (RF)
  - k) Medição de descarga parcial
  - l) Análise de assinatura de corrente do motor (MCSA)
  - m) Análise estatística de sinais relevantes
- 2) Diagnóstico de falha baseado em modelo
  - a) Rede neural
  - b) Análise de lógica difusa

- c) Algoritmo genético
  - d) Inteligência artificial
  - e) Equivalentes de circuito magnético de elemento finito (FE)
  - f) Modelos matemáticos baseados na teoria do circuito linear
- 3) Análise de falhas baseada na teoria da máquina
- a) Abordagem da função de enrolamento (WFA)
  - b) Abordagem da função de enrolamento modificada (MWFA)
  - c) Circuito magnético equivalente (MEC)
- 4) Análise de falhas baseada em simulações
- a) Análise de elementos finitos (FEA)
  - b) Análise de espaço de estado de elemento finito acoplado por etapa de tempo (TSCFE-SS)

Com base em tudo exposto até aqui, o presente trabalho se propõe a apresentar uma forma de detecção de algumas falhas baseado em modelo (grupo 2 de TOLİYAT) com o uso de dados sonoros.

## 1.1 OBJETIVOS

O presente trabalho tem objetivo geral e objetivos específicos.

### 1.1.1 Objetivo Geral

O objetivo geral é a criação de um protótipo funcional que detecte falhas baseado em dados sonoros bem como descrever uma metodologia de projeto de algoritmos de aprendizado de máquina.

Além disso, exploram-se técnicas de análise e tratamento de dados que visam garantir que as especificações de saída sejam satisfeitas servindo como um manual geral para futuros projetos e tornando este documento uma referência para futuros projetistas.

### 1.1.2 Objetivos Específicos

Os objetivos específicos englobam os seguintes tópicos:

- Definir as fases de um projeto de Aprendizado de Máquina.



- Entender o que é um algoritmo de aprendizado de máquina e o seu funcionamento básico.
- Demonstrar como se obtém dados e como se tratar os mesmos.
- Prover formas de avaliar e escolher o melhor algoritmo para cada situação.
- Demonstrar como medir a performance de um algoritmo de aprendizado de máquina.
- Entender como usar um algoritmo de aprendizado de máquina em um caso real.

## 1.2 METODOLOGIA

O presente trabalho se caracteriza como um estudo experimental pois visa determinar um objeto de estudo, selecionar e discutir as variáveis capazes de influenciá-lo além de definir as formas de controle e de observação dos efeitos que cada variável produz no objeto.

Como o desenvolvimento e construção de um sistema de detecção baseado em aprendizado de máquina é longo e complexo, será utilizada uma abordagem sistemática para facilitar o entendimento e prover uma forma genérica para construção prática de sistemas baseados nesta técnica.

Para este trabalho, foram coletados dados de um motor elétrico com potência 135W com o uso um microfone. Estes dados foram armazenados no formato “.wav” e, posteriormente, os dados foram analisados e usados para treinar um modelo de aprendizado de máquina. Por fim, este modelo foi usado para fazer a análise em tempo real do equipamento a fim de detectar falhas apenas com base no som produzido pelo mesmo.

## 2 APRENDIZADO DE MÁQUINA

Aprendizado de máquina é um termo que tem ganhado notoriedade na atualidade, porém, de forma geral, associado a um tom de misticismo ou desconhecimento fortemente corroborado pela mídia. Para evitar confusão ou misticismo, bem como prover uma forte base para o desenvolvimento de futuros projetos, a primeira parte deste trabalho demonstra os principais aspectos referentes a este tema.

### 2.1 O QUE É APRENDIZADO DE MÁQUINA?

O significado de Aprendizado de máquina, do inglês *machine learning*, pode ser encontrado na ENCICLOPÉDIA sendo definido como: “Um subcampo da Engenharia e da ciência da computação que evoluiu do estudo de reconhecimento de padrões e da teoria do aprendizado computacional em inteligência artificial”.

Na visão do autor do presente trabalho, uma das melhores classificações de aprendizado de máquina é trazida por CHOLLET, 2017, pg. 5, na qual ele define aprendizado de máquina como “um novo paradigma de programação”.

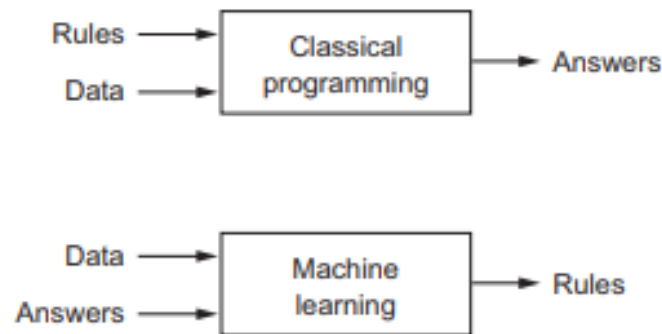
#### 2.1.1 Programação tradicional vs Aprendizagem de máquina

A forma mais fácil de se explicar este conceito é fazer um paralelo com um programa de computador tradicional.

Um programa de computador tradicional pode ser resumido em um conjunto de dados e regras que produzem um resultado, ou seja, escreve-se um script e este programa segue as regras deste script ao ser executado.

Já em um programa de aprendizado de máquina, se entrega as entradas e os resultados esperados e as regras são geradas pelo programa.

Figura 1 – Aprendizado de máquina: um novo paradigma de programação



Fonte: CHOLLET, 2017, pg. 5

Por exemplo, ao entregar as características de um conjunto de imóveis (dados) e o preço esperado em cada um deles (saídas esperadas), o algoritmo deve ser capaz de, com as características de um novo imóvel, predizer o seu preço exclusivamente a partir de suas características.

### 2.1.2 Como o algoritmo sabe a resposta certa?

Uma dúvida pode surgir ao trabalhar com um algoritmo de Aprendizado de Máquina é: “como ele sabe qual é a resposta certa?”.

Para que um algoritmo possa prover uma resposta adequada é necessário treiná-lo, ou seja, ajustar o seu modelo a fim de enquadrar o problema.

Existem diversos modelos e formas de treinamento os quais pode-se citar:

- Regressão Linear do inglês *Linear Regression*.
- Máquinas de Vetor de Suporte (MVS) do inglês Support Vector Machines (SVM).
- Gradiente Descendente (GD) e suas variações de treinamento Gradiente Descendente em Lotes (GDL) ou Gradiente Descendente Estocástico (GDE).
- Regressão Logística.
- Redes Neurais do inglês *Neural Networks*.

Este trabalho não tem foco em explicar como cada modelo funciona ou é treinado, entretanto, é importante desmistificar o treino de um modelo e prover uma explicação breve ao leitor. Para explicar como funciona o processo de treino, um exemplo simples será demonstrado usando Regressão Linear.

Tomando por base a Tabela 1 que contém os dados do PIB (GDP) per capita e da satisfação de vida de alguns países extraídos do site *OUR WORLD DATA* pode-se traçar os seguintes questionamentos:

- Será que um GDP per capita maior é refletido em uma maior satisfação de vida?
- Como estimar a satisfação de vida de um país com base apenas no dado de GDP per capita?

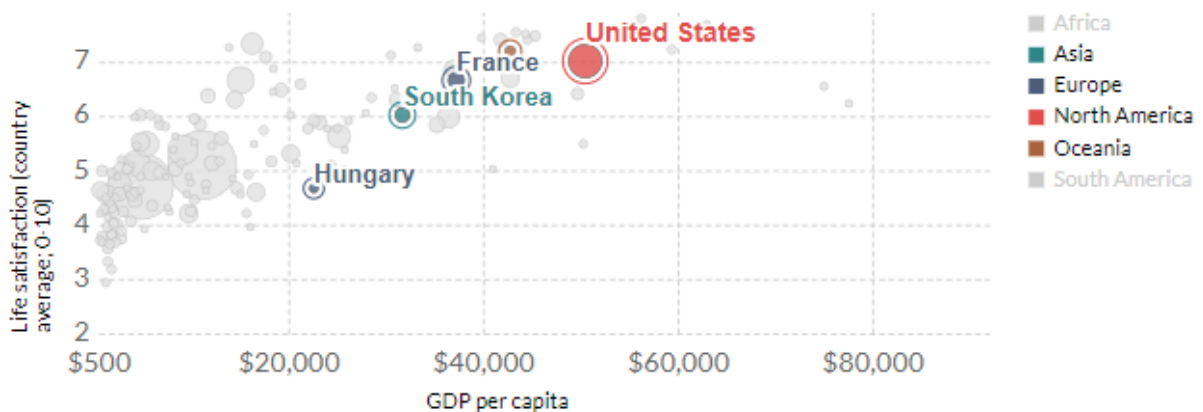
Tabela 1 – GDP per capita e Satisfação de vida por país – 2021

País	GDP per capita (USD)	Satisfação de vida (Média do país; 0-10)
Hungria	\$ 26,778	6.07
Coréia do Sul	\$ 35,938	5.84
França	\$ 38,606	6.67
Austrália	\$ 44,649	7.18
Estados Unidos	\$ 54,225	6.88

Fonte: Our World Data (2021)

Para expor melhor os dados a Figura 2 foi criada.

Figura 2 – Satisfação de vida vs GDP



Fonte: Our World Data (2021)

Na Figura 2, pode-se observar uma correlação entre o aumento da satisfação de vida com o aumento do GDP. Desta forma, pode-se aproximar esta correlação com um modelo linear de reta como descrito na Equação 1.

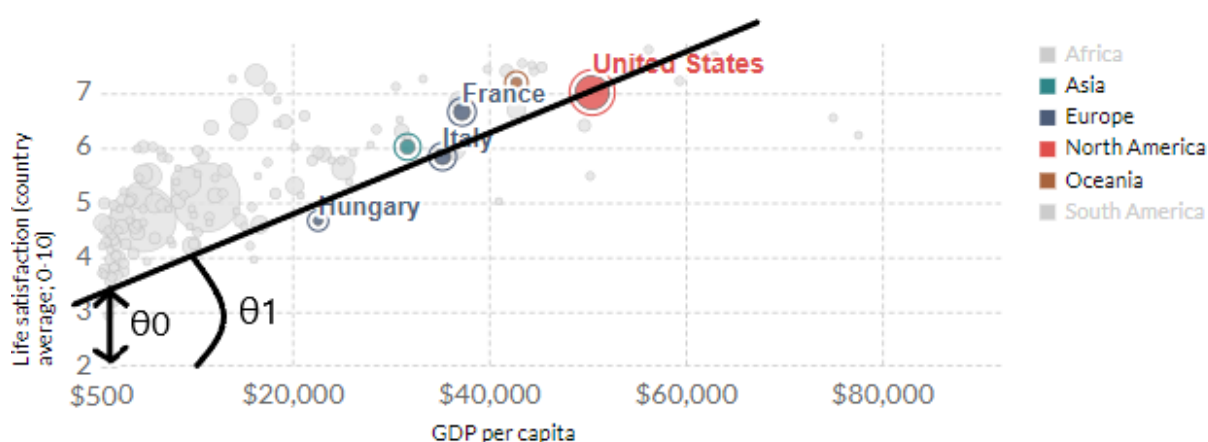
$$Satisfacaodevida = \theta_0 + \theta_1 * GDPpercapita \quad (1)$$

É fácil ver que diferentes valores de  $\theta_0$  e  $\theta_1$  provocam uma aproximação diferente dos dados na reta.

O trabalho central do algoritmo de Aprendizado de Máquina é encontrar a reta que melhor se ajusta aos dados. Para isso, o algoritmo usa uma métrica de erro como, por exemplo, a média da diferença que existe entre cada um dos pontos até a reta (Erro médio absoluto).

E qual o papel desta reta? Com este modelo linear ajustado, pode-se agora, prever valores que não estão no conjunto de dados como por exemplo o valor para a Itália observado na Figura 3.

Figura 3 – Regressão linear da Satisfação de vida vs GDP



Fonte: Acervo do Autor (2021)

Claramente, existem diversos outros algoritmos, formas de treinamento e aplicações, entretanto, este exemplo mostra de forma simplificada como um algoritmo pode ser treinado para gerar informação útil.

### 3 FASES DE UM PROJETO DE APRENDIZADO DE MÁQUINA

Com os conceitos básicos de Aprendizado de máquina estabelecidos, agora é necessário entender como um projeto é conduzido. AURELIEN, 2017, pg. 497-498, sugere um checklist de desenvolvimento com base nas seguintes etapas:

1. Observe o quadro geral.
2. Obtenha os dados.
3. Descubra e visualize os dados para obter insights.
4. Prepare os dados para algoritmos de aprendizado de máquina.

5. Selecione um modelo e treine-o.
6. Ajuste seu modelo.
7. Apresente sua solução.
8. Inicie, monitore e mantenha seu sistema.

É preciso ter em mente que este é um checklist e não um roteiro genérico visto que um projeto de aprendizado de máquina não possui uma forma única de ser executado, nem regras pétreas sobre as fases que devem ser feitas ficando a cargo da experiência de cada desenvolvedor demonstrar a melhor forma para cada problema.

Cada uma destas fases será observada em mais detalhes a seguir usando um paralelo com o problema central deste trabalho.

### 3.1 DEFINIÇÃO DO PROBLEMA

AURELIEN, 2017, pg. 35 “Como a empresa espera usar e se beneficiar desse modelo? Isso é importante porque vai determinar como você enquadra o problema, quais algoritmos você seleciona, que medida de desempenho você usará para avaliar seu modelo e quanto esforço você deve gastar para ajustá-lo.”.

Como observado na citação, é necessário entender o problema e as suas implicações respondendo perguntas como:

- Quais são os problemas comparáveis?
- É possível reutilizar experiências ou ferramentas já desenvolvidas?
- Existe experiência humana disponível?
- Como este problema é resolvido manualmente?

Com base em nestas respostas o rumo do projeto fica mais claro. Para exemplificar, o Quadro 1 foi gerado para enquadrar o problema proposto neste trabalho.

Quadro 2 – Enquadramento do problema

Pergunta	Resposta	Justificativa
Tipo de algoritmo	Classificador	Saídas categóricas
Métrica de qualidade	F1 Score	Uma medida balanceada para a medição de erro
Desempenho aceitável	75% de F1 Score para as classes de Falta de tensão e Desbalanceamento	O correto seria analisar o quão preciso as atuais

	do eixo.	técnicas são e então traçar, junto aos líderes de manutenção quais as classes prioritárias e a precisão aceitável, pela ausência destes dados, optou-se por um valor de 75% visto que é 3 vezes acima da aleatoriedade (25%).
Quais os problemas comparáveis e é possível reutilizar experiências ou ferramentas já desenvolvidas?	Pode-se tomar como base FRIZZO STEFENON e SCHMIDT	
Existe experiência humana disponível?	Não aplicável.	Aqui seria interessante conversar com o gestor de manutenção e verificar se ele tem alguma experiência que pode ser útil na obtenção e exploração dos dados como por exemplo a frequência que a máquina opera e quais os parâmetros da mesma.
Como este problema é resolvido manualmente?	Existem técnicas de análise periódicas definidas em ELECTRIC MACHINES.	

Como é possível observar, esta fase já implica muito nas próximas etapas. Com as perguntas iniciais sanadas, o projeto pode seguir para a próxima etapa.

### 3.2 OBTENÇÃO DOS DADOS

Nesta etapa, a quantidade e o tipo dos dados são levantados. Também é decidido de onde os dados serão extraídos e onde os dados serão armazenados.

Outro fator importantíssimo é a legalidade da extração dados, como HUMBY, 2013, pg.1, disse “Os dados são o novo petróleo” e, portanto, devem ser protegidos e extraídos apenas de fontes legais e com as devidas permissões dos possíveis envolvidos.

Traçando um paralelo com o problema deste trabalho, é preciso obter dados sonoros de um equipamento, portanto, é necessária uma consulta ao setor de manutenção e gerência para obter as devidas autorizações. Os dados devem ser armazenados em um formato de áudio como “mp3” ou “wav” e salvos em uma estrutura de nomes ou diretórios facilmente identificáveis por classe.

### 3.3 EXPLORAÇÃO DOS DADOS

Segundo MCKINNEY, 2017, pg. 190, “Durante o curso de análise e modelagem de dados, uma quantidade significativa de tempo é gasta na preparação de dados: carregamento, limpeza, transformação e reorganização”. Este tempo é muito importante pois padrões importantes são detectados e dados inúteis são descartados.

Em resumo, nesta fase, os dados são efetivamente tratados. Os tratamentos são diversos e incluem técnicas como a remoção ou tratamento de dados nulos, remoção de variáveis desnecessárias e técnicas de engenharia de recursos como a agregação ou decomposição de colunas além dos processos de escalonamento dos dados.

Ainda nesta etapa é necessário verificar o tipo dos dados, processar textos, sons, imagens e variáveis categóricas para valores numéricos que podem ser usados por um algoritmo de aprendizado de máquina.

Ferramentas como a Matriz de correlação e gráficos dos mais diversos são ótimos para “torturar os dados” e descobrir padrões nos mesmos.

No problema a ser resolvido neste trabalho, pode-se usar técnicas como a plotagem das formas de onda captadas, verificar se existem dados fora de escala ou medições incorretas ou até mesmo aplicar transformações nos dados em verificar qual é o seu comportamento.

Esta é uma etapa que pode ser repetida várias vezes até encontrar o melhor processo.

### 3.4 PREPARAÇÃO DOS DADOS PARA INCLUIR NO ALGORITMO

Como exposto por MCKINNEY, 2017, pg. 60-65, a maioria dos algoritmos de aprendizado de máquina não funciona com recursos ausentes ou fora de escala e prefere trabalhar com números ao invés de textos.

Com base nestas premissas, esta etapa divide os dados em conjuntos de treino e teste, verifica e elimina dados nulos, cria o escalonamento de dados e provê um conjunto de dados que o algoritmo consiga usar para ser treinado.

### 3.5 VALIDAÇÃO DO MELHOR MODELO

Após o tratamento, é necessário validar se as modificações causaram efeitos positivos, para isso, a melhor forma é treinar diversos algoritmos e verificar como eles se comportam frente à novas predições.



Novamente fazendo um paralelo com o problema, cada técnica utilizada nos dados nas etapas anteriores pode ser aplicada usando algoritmos diferentes. Com isso, pode-se verificar qual delas oferece o menor erro desejado.

### 3.6 AJUSTE E COMBINAÇÃO DE MODELOS

Após a validação dos dados, um candidato de processo que entregue a melhor forma para tratá-los é encontrado. Com este candidato em mãos, é necessário treinar diversos modelos, combiná-los e buscar o melhor ajuste através da parametrização para o problema em questão. Nesta fase, se faz uso de técnicas como a busca em grade e a união de vários algoritmos em uma solução única.

Para atrelar esta fase ao projeto central deste trabalho, basta testar vários modelos e usar a busca em grade para encontrar o melhor ajuste.

### 3.7 APRESENTAÇÃO E DOCUMENTAÇÃO DA SOLUÇÃO

Apesar de ser uma fase abdicada por muitos desenvolvedores e cientistas, deve-se documentar tudo que foi feito desde a obtenção dos dados até a colocação do modelo em produção pois, desta maneira, pode-se usar a experiência obtida em projetos futuros bem como ter um registro histórico para rever o projeto no futuro.

No presente trabalho a maneira adotada é a utilização de um documento JUPYTER no qual códigos PYTHON, tabelas HTML, imagens e diversos outros elementos podem ser inseridos de forma dinâmica provendo uma documentação riquíssima e um laboratório de dados poderosíssimo.

### 3.8 IMPLANTAÇÃO E MONITORAMENTO DO SISTEMA

“Você precisa preparar sua solução para produção, em particular conectando as fontes de dados de entrada de produção em seu sistema e escrevendo testes.” (MCKINNEY, 2017, pg. 76).

Ou seja, com os dados estudados, a melhor forma de tratá-los encontrada e o modelo desenvolvido e treinado, está na hora de integrá-lo a fonte geradora de dados e monitorar os resultados.

Como explicitado por MCKINNEY, 2017, pg. 76, nesta etapa, é necessário ter em mente que um sistema nunca está finalizado e é necessário verificar a sua performance com novos

dados, monitorar sua eficiência e, se necessário, dar manutenção a fim de trazer melhor eficácia. Ou seja, não basta apenas ligar o sistema a fonte geradora de dados pois todo sistema sofre degradação do tempo (um sensor com falha ou uma mudança no processo por exemplo) e pode ser melhorado a fim de atingir melhor eficiência.

No presente trabalho, a forma mais conveniente de integrar o sistema é exportando o modelo treinado e criar um novo algoritmo que usa este modelo em leituras em tempo real. E, para fazer o monitoramento pode-se criar uma interface gráfica.

## 4 PROJETO PRÁTICO

Passado uma introdução teórica acerca do assunto, na presente seção do trabalho será abordado de forma prática e detalhada como o projeto do protótipo foi concebido.

### 4.1 COLETA DOS DADOS

Para o projeto, duas situações foram simuladas, uma denominada “som ambiente”, na qual apenas o som do equipamento foi gravado e outra denominada “som industrial inserido” na qual o som referenciado como INDUSTRY foi inserido por um autofalante próximo da região de captação.

#### 4.1.1 O setup de coleta

Em ambas as situações, o setup da Figura 4 foi usado. Neste setup, um microfone modelo BM800 é usado como sensor sonoro. Para otimizar a captação, o microfone foi conectado a uma fonte de tensão 48V do tipo Phantom Power.

Através de um cabo, a captação é conectada a um Notebook com 8 Gigabytes de memória RAM, 1TB de SSD, um processador i5 rodando o sistema operacional Ubuntu 20.4, claramente sobre dimensionado para a tarefa.

Figura 4 – Setup para a captação de dados



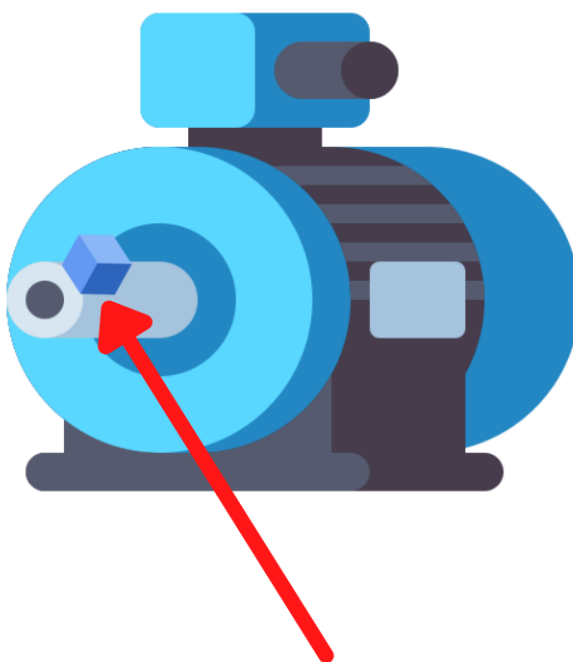
Fonte: Acervo do autor (2021)

#### 4.1.2 Situações simuladas

No presente trabalho, alguns dos defeitos citados na introdução foram simulados. Por questão de conveniência e facilidade de simulação as seguintes condições foram gravadas:

- Regime normal, ou seja, com o equipamento funcionando perfeitamente.
- Falta de tensão, ou seja, gravando com o equipamento desligado da alimentação.
- Sobrecarga, nesta condição o equipamento foi submetido a uma carga superior a suportada.
- Desbalanceamento de eixo, uma pequena massa de aproximadamente 5g foi inserida no eixo do equipamento (como observado na Figura 5) causando o desbalanceamento de eixo.

Figura 5 – Defeito de desbalanceamento de eixo simulado



Inserção de massa

Fonte: Acervo do autor (2021)

### 4.1.3 A linguagem utilizada

RAMALHO, 2015, p.28, “Uma das melhores qualidades de PYTHON é a sua consistência.”. Além da consistência da linguagem por questões de facilidade e familiaridade do autor a linguagem PYTHON foi escolhida para criar o algoritmo de gravação dos dados.

Ainda é importante citar o módulo PYAUDIO muito importante para a obtenção dos dados. O algoritmo usado está disponível em DETECTOR e no Apêndice A deste documento.

Os arquivos foram salvos no formato “.wav” pela conveniência da linguagem utilizada em gerar estes arquivos. Ao salvar os dados, cada arquivo recebeu a data e hora da gravação em seu nome.

Com base em todas as informações a seguinte estrutura de diretórios e arquivos foi gerada:

```
.
├── Saídas\
│   ├── 0 - Regime Normal\
│   │   └── 2021-05-26 20:52:05.056246.wav (arquivo de áudio gravado)
│   ├── 1 - Falta de tensão\
│   │   └── 2021-05-26 21:52:05.056246.wav (arquivo de áudio gravado)
│   ├── 2 - Sobrecarga\
│   │   └── 2021-05-26 22:52:05.056246.wav (arquivo de áudio gravado)
│   ├── 3 - Desbalanceamento do eixo\
│   │   └── 2021-05-26 20:53:05.056246.wav (arquivo de áudio gravado)
│   └── main.py (algoritmo para gravação dos dados)
```

#### 4.1.3.1 Quantos dados devem ser gravados?

Na etapa de coleta de dados a pergunta mais difícil de ser respondida é: “qual é a quantidade de dados que devem coletados?”

Segundo ALEXANDRE, 2019, pg. 1, “Todos os projetos são de alguma forma únicos, mas eu diria que você precisa de 10 vezes mais dados do que o número de parâmetros no modelo que está sendo construído. Quanto mais complicada for a tarefa, mais dados serão necessários”.

Entretanto esta é uma abordagem muito genérica e para entender melhor o que está por trás disso BROWNLEE, 2017, pg. 1, demonstra as principais problemáticas e uma abordagem mais rica para estimar a necessidade da quantidade de dados necessária.

Em resumo, diversos fatores como a complexidade do problema, a dificuldade de treinamento do algoritmo, a quantidade de variáveis de entrada e a quantidade de classes de saída tornam esta tarefa puramente heurística e prática.

Para o problema específico do presente trabalho, o principal desafio está nas condições que os dados podem ser extraídos. Visando apenas o ambiente de amostragem montado, um número potencialmente infinito de amostras pode ser coletado visto que os dados são facilmente obtidos. Entretanto, para tornar o exemplo mais tangível à realidade das empresas, apenas 12 amostras de cada falha serão coletadas para cada um dos cenários de ambiente.

## 4.2 EXPLORAÇÃO DOS DADOS

MCKINNEY, 2017, pg. 250 diz “Fazer visualizações informativas (às vezes chamadas de gráficos) é uma das tarefas mais importantes na análise de dados”. Esta parece ser uma afirmação forte, mas é nesta fase que os gráficos são criados para tentar expor padrões e tornar a análise mais compreensível.

Para facilitar o trabalho, os dados foram carregados em um DataFrame da biblioteca pandas.

Um DataFrame representa uma tabela retangular de dados e contém uma coleção ordenada de colunas, cada uma das quais pode ser um tipo de valor diferente (numérico, string, booleano, etc.). O DataFrame tem um índice de linha e coluna; pode ser pensado como um dicionário de Series, todos compartilhando o mesmo índice. Sob o capô, os dados são armazenados como um ou mais blocos bidimensionais em vez de uma lista, dicionário ou alguma outra coleção de matrizes unidimensionais. (MCKINNEY, 2017, pg. 128).

Neste conjunto, cada gravação de áudio se torna a linha de uma tabela e cada coluna reflete o valor absoluto de intensidade de áudio no momento que foi gravado. A Figura 6 demonstra esta estrutura.

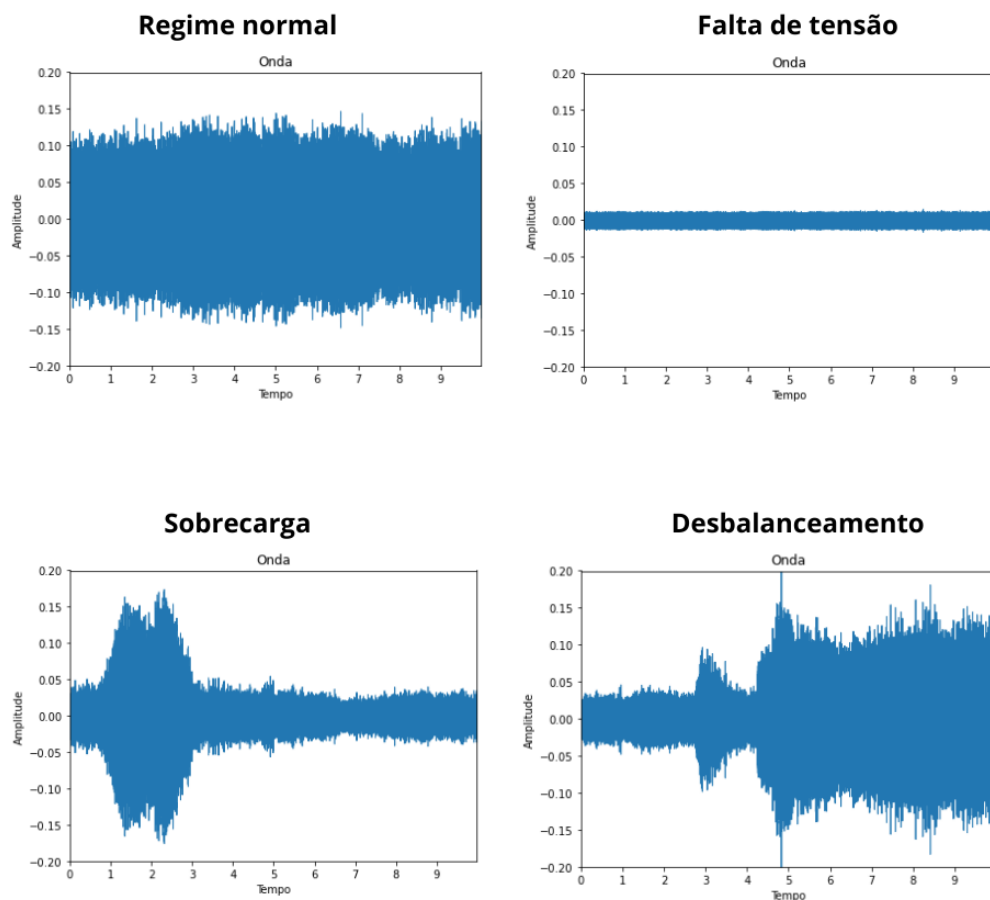
Figura 6 – DataFrame com os dados carregados

	0	1	2	3	4	5	6	7	8	9	...	2039	2040	2041	2042	2043	2044	2045	2046
0	3013.0	3016.0	3099.0	3115.0	3074.0	3058.0	3047.0	2989.0	2858.0	2727.0	...	998.0	946.0	885.0	837.0	728.0	586.0	487.0	406.0
1	-551.0	-399.0	-394.0	-418.0	-445.0	-379.0	-283.0	-310.0	-253.0	-125.0	...	2240.0	2329.0	2438.0	2558.0	2550.0	2380.0	2297.0	2343.0
2	-1305.0	-1221.0	-1244.0	-1234.0	-1137.0	-1079.0	-1059.0	-982.0	-876.0	-825.0	...	1160.0	1007.0	991.0	1121.0	1205.0	1166.0	1145.0	1054.0
3	2893.0	2965.0	3006.0	3076.0	3173.0	3177.0	3174.0	3161.0	3144.0	3099.0	...	3965.0	3978.0	4080.0	4123.0	4145.0	4037.0	3912.0	3832.0
4	2305.0	2196.0	2092.0	2028.0	1909.0	1807.0	1776.0	1839.0	1902.0	1943.0	...	-1234.0	-1330.0	-1386.0	-1403.0	-1380.0	-1344.0	-1329.0	-1344.0
5	1775.0	1653.0	1672.0	1650.0	1670.0	1720.0	1767.0	1847.0	1947.0	2004.0	...	3742.0	3800.0	3850.0	4008.0	4176.0	4197.0	4116.0	4012.0

Fonte: Acervo do autor (2021)

Números, entretanto, são difíceis de se observar, com base nisso, um dos gráficos gerados é o de forma de onda representado na Figura 7 comparando cada classe do conjunto de dados e tornando a visualização da amplitude do som gravado mais fácil.

Figura 7 – Formas de onda para cada classe



Fonte: Acervo do autor (2021)

Em geral, em algoritmos de classificação se busca detectar o que diferencia cada classe alvo. Como por exemplo, em classificadores de imagem as cores, formas, ou pixels que definem melhor cada tipo de imagem.

Ao observar a Figura 7, pode-se supor que valores como a amplitude média ou mediana, os valores de máximos e mínimos podem ser úteis para distinguir classes visto que é diferente em cada classe.

Para testar estes valores bastaria manipular os dados a fim de extrair estes valores como observado na Figura 8.

Figura 8 – DataFrame com os dados carregados

	<b>Média</b>	<b>Minímo</b>	<b>Máximo</b>
<b>0</b>	556.007812	-3767.0	3994.0
<b>1</b>	296.060547	-3701.0	4180.0
<b>2</b>	268.788574	-4023.0	4593.0
<b>3</b>	650.110840	-4091.0	4305.0
<b>4</b>	313.874023	-3681.0	4748.0

Fonte: Acervo do autor (2021)

Além destas técnicas básicas, diversas outras formas de processamento e manipulação dos dados podem ser usadas e criadas como apontado por UNPINGCO.

Uma técnica que é comumente utilizada em processamento de áudio e que possui uma implementação simples em PYTHON é a transformada de Fourier.

A análise de Fourier é um método para expressar uma função como uma soma de componentes periódicos e para recuperar o sinal desses componentes. Em processamento digital, quando a função e sua transformada de Fourier são substituídas por contrapartes discretizadas, ela é chamada de transformada discreta de Fourier (DFT) descrita na Equação 2.

$$X(k) = \sum_{n=0}^{N-1} e^{-2\pi j \frac{kn}{N}} * x(n) \quad (2)$$

Onde:

$X$  é uma sequência de números complexos do espectro

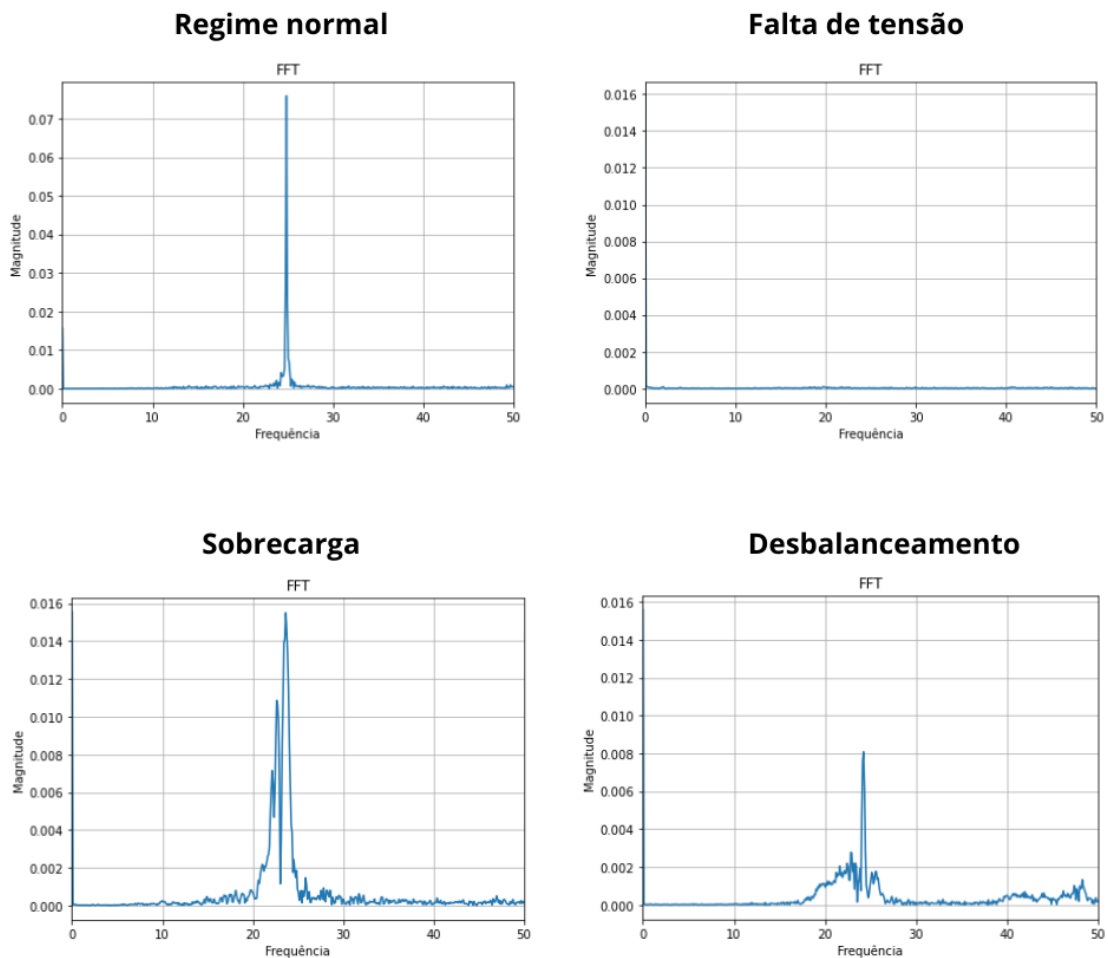
$x$  é o vetor contendo as amostras discretas das amplitudes



Para o presente trabalho, com dados discretos, existe um poderoso algoritmo conhecido como Transformada Rápida de Fourier do inglês “Fast Fourier Transform (FFT)”.

Após a aplicação deste algoritmo, a Figura 9 pode ser gerada demonstrando padrões interessantes.

Figura 9 – FFT para as diferentes classes

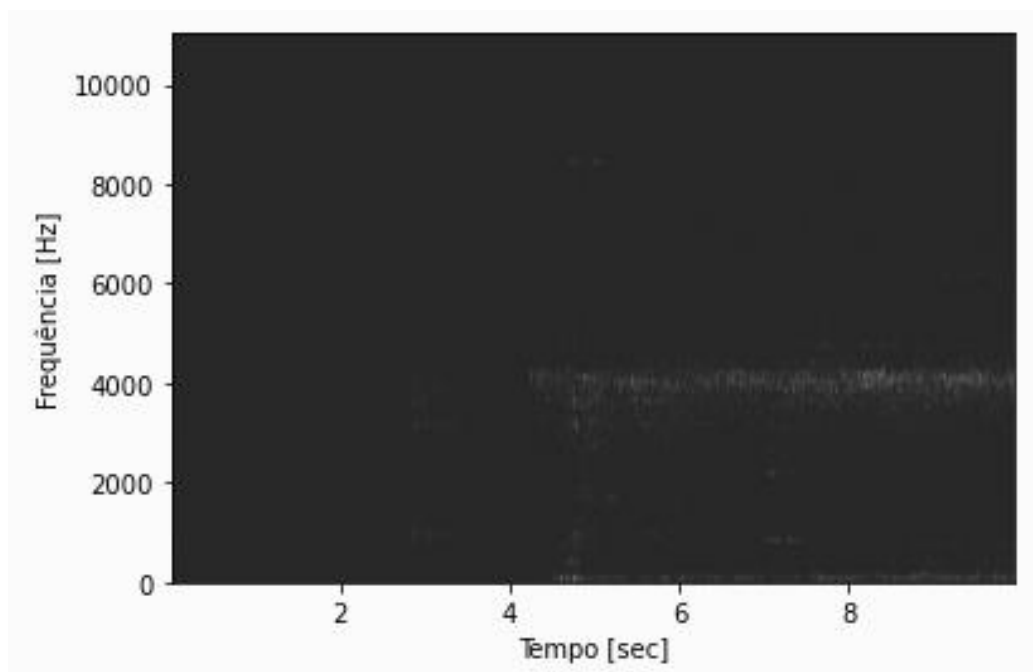


Fonte: Acervo do autor (2021)

Com este gráfico pode-se levantar a suposição que aplicar a transformada de Fourier e trabalhar com os dados no domínio da transformada ajuda a diferenciar as classes visto que os espectros de frequência estão diferentes.

Um outro método gráfico muito interessante em dados de áudio é o espectrograma, no qual representa a frequência frente ao tempo como demonstrado na Figura 10 para o desbalanceamento de classes.

Figura 10 – Espectrograma para eixo desbalanceado



Fonte: Acervo do autor (2021)

Como observado, a frequência possui variação expressiva em algumas amostras, porém não possui um padrão muito visível e constante nestas variações.

### 4.3 TRANSFORMAÇÃO

A transformação dos dados está ocorrendo em cada uma das fases expostas até o momento. Como por exemplo na captação dos dados onde uma representação digital do som foi armazenada e na aplicação da transformada de Fourier em cada gravação dos dados.

Com base nisso, é necessário ter em mente que algumas etapas podem se mesclar, mudar a sua ordem ou ser aplicadas diversas vezes para se obter o resultado esperado.

Algumas transformações e explorações foram omitidas aqui, mas é fortemente recomendável ver o DETECTOR e observar as demais formas de processamento utilizadas.

### 4.4 TESTE E VALIDAÇÃO DO MODELO

Com as suposições criadas, é hora de validar a melhor forma de transformar os dados. Para isso, a forma mais adequada é aplicar as transformações nos dados e testá-los em diversos modelos para obter as hipóteses mais promissoras.

Como diversas alternativas foram testadas é indicado conferir o DETECTOR e observar os arquivos referentes a análise dos dados para todas as hipóteses. Para fins práticos e para evitar a repetição apenas o melhor processo encontrado para este caso será demonstrado.

#### 4.4.1 Transformação dos dados

De todas as formas testadas, a função descrita na Figura 11 foi usada, para cada registro de som no conjunto de dados, a fim de obter o melhor resultado.

Figura 11 – Função de transformação dos dados

```
def transform_array(arr):
    """ Esta função retorna os valores médios, mínimos,
    máximos e a média das componentes real e imaginária da FFT """

    mean = arr.mean()
    minimun = arr.min()
    maximun = arr.max()

    fft_value = np.fft.fft(arr)
    real_part = np.abs(fft_value).mean()
    imaginary_part = np.imag(fft_value).mean()

    return np.array([mean, minimun, maximun, real_part, imaginary_part])
```

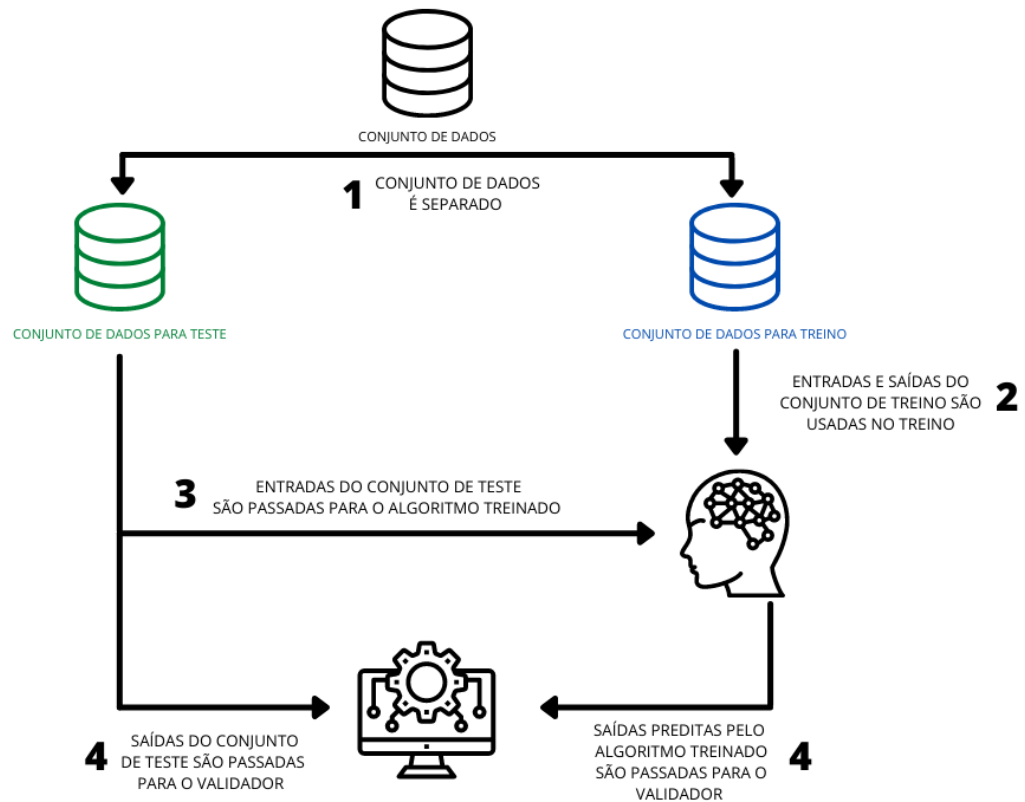
Fonte: Acervo do autor (2021)

Nesta função, os valores de máximo, mínimo, média e a média das partes real e imaginária da FFT de cada registro são retornados.

##### 4.4.1.1 Separação dos conjuntos de treino e teste

A fim de saber o quão precisos os algoritmos treinados são, um subconjunto dos dados transformados foi separado para fazer a validação. Este conjunto foi obtido em uma razão de 30% de teste e 70% de treino. A Figura 12 demonstra como é usado este conjunto.

Figura 12 – Fluxo de treino e validação dos modelos



Fonte: Acervo do autor (2021)

#### 4.4.2 Treino e validação dos algoritmos

Com os conjuntos separados, é necessário treinar diversos algoritmos e obter o que melhor se adequa aos dados.

Para medir o resultado existem diversas métricas e ferramentas, no projeto foram utilizadas as seguintes:

- Matriz de confusão
- Recall
- Precisão
- F1 Score

##### 4.4.2.1 Matriz de confusão

A Matriz de confusão é uma ferramenta que possibilita ver os erros que um algoritmo cometeu ao classificar registros. Ou seja, ver quantas vezes uma classe qualquer A foi classificada como outra classe qualquer B.

Seu funcionamento é simples, os dados reais (do conjunto de teste) são confrontados com as predições (feitas pelo algoritmo) e uma tabela NxN (sendo N o número de classes de saída) evidenciando, desta forma, o número de acertos e erros em cada classe.

Uma Matriz de confusão perfeita ocorre quando todos os seus valores estão na diagonal, como demonstrado na Tabela 2.

Tabela 2 – Matriz de confusão ideal

		Conjunto predito (Pelo algoritmo)			
		Regime Normal	Falta de Tensão	Sobrecarga	Desbalanceamento de Eixo
Conjunto de testes (Valores reais)	Regime Normal	2	0	0	0
	Falta de Tensão	0	4	0	0
	Sobrecarga	0	0	1	0
	Desbalanceamento de Eixo	0	0	0	3

Fonte: Acervo do autor (2021)

No entanto, na maioria das vezes, não é possível obter uma Matriz de confusão perfeita. Para o presente estudo, um dos melhores resultados obtidos está explicitado na Tabela 3.

Tabela 3 – Matriz de confusão real

		Conjunto predito (Pelo algoritmo)			
		Regime Normal	Falta de Tensão	Sobrecarga	Desbalanceamento de Eixo
Conjunto de testes (Valores reais)	Regime Normal	1	0	1	0
	Falta de Tensão	0	2	2	0
	Sobrecarga	0	0	1	0
	Desbalanceamento de Eixo	0	0	0	3



Fonte: Acervo do autor (2021)

Para entender esta matriz, observa-se os dados por linhas e colunas. A primeira coluna e a primeira linha refletem os valores da primeira classe (Regime normal). Para esta classe, pode-se ver que o algoritmo classificou corretamente 1 registro como Regime Normal (Verdadeiro Positivo) e incorretamente 1 registro como Sobrecarga sendo que deveria ser Regime Normal (Falso Negativo). Podemos ainda, observando a primeira coluna, notar que o algoritmo

não classificou incorretamente as outras classes como Regime Normal, ou seja, ele não cometeu Falsos Positivos. Tudo isso está indicado na Figura 13.

Figura 13 – Matriz de confusão com dados destacados

		Conjunto predito (Pelo algoritmo)			
		Regime Normal	Falta de <u>Tensão</u>	Sobrecarga	<u>Desbalanceamento</u> de <u>Eixo</u>
Conjunto de testes (Valores reais)	Regime Normal	1	0	1	0
	Falta de <u>Tensão</u>	0	2	2	0
	<u>Sobrecarga</u>	0	0	1	0
	<u>Desbalanceamento</u> de <u>Eixo</u>	0	0	0	3

 Corretamente classificados
  Incorretamente classificados

Fonte: Acervo do autor (2021)

Na segunda coluna e segunda linha a classe em questão é Falta de Tensão. Nesta classe pode-se observar que 2 registros foram classificados corretamente como Falta de Tensão (Verdadeiro Positivo) e 2 registros incorretamente como Sobrecarga sendo que na verdade são Falta de Tensão (Falso Negativo). Pode-se ainda, observando a segunda coluna, constatar que não houve nenhum registro de outra classe sendo incorretamente classificado como Falta de Tensão, desta maneira, pode-se afirmar que não existem Falsos Positivos para esta classe.

Já na terceira coluna e terceira linha um registro foi corretamente classificado como Sobrecarga (Verdadeiro Positivo) e nenhum registro foi incorretamente enquadrado nesta categoria. Entretanto, ao observar a coluna, pode-se constatar que 2 registros de Falta de Tensão e 1 registro de Regime normal foram classificados incorretamente como Sobrecarga, ou seja, 3 falsos positivos.

De maneira análoga, a quarta coluna possui 3 amostras corretamente classificadas em Desbalanceamento de Eixo (Verdadeiro Positivo) e nenhuma outra classe foi enquadrada incorretamente como Desbalanceamento de eixo. Também é possível notar que não existem falsos negativos nesta classe.

Entender o funcionamento da Matriz de confusão é um passo vital visto que as demais métricas usadas aqui derivam dela.

#### 4.4.2.2 Precisão

A Matriz de Confusão prove uma série de informações, entretanto, uma métrica absoluta é necessária. A primeira métrica absoluta analisada foi a precisão.

Para determinar a precisão do algoritmo em cada classe, se usa a Equação 4:

$$Precisão = \frac{VerdadeirosPositivos}{VerdadeirosPositivos+FalsosPositivos} \quad (4)$$

Para a primeira classe, temos:

$$Precisão = \frac{1}{1+0} = 1 \quad (5)$$

Desta maneira, pode-se afirmar que, para o conjunto de testes, o algoritmo possui 100% de precisão na detecção de “Regime Normal”, ou seja, todos os registros classificados como “Regime Normal” são verdadeiramente “Regime Normal”.

A mesma equação pode ser aplicada para as demais classes.

#### 4.4.2.3 Recall

Uma outra métrica muito interessante é o Recall, essa métrica atribui a taxa de registros positivos classificados corretamente. A Equação 6 descreve como se obtém o recall.

$$Recall = \frac{VerdadeirosPositivos}{VerdadeirosPositivos+FalsosNegativos} \quad (6)$$

Para a primeira classe, temos:

$$Recall = \frac{1}{1+1} = 0.5 \quad (7)$$

Desta forma, pode-se afirmar que, para o conjunto de testes, o algoritmo possui 50% de Recall na primeira classe. Ou seja, 50% de “Regime Normal” foi incorretamente classificado em outra classe (neste caso particular, 1 registro foi classificado como “Sobrecarga”).

A mesma fórmula pode ser aplicada para as demais classes.

#### 4.4.2.4 F1 Score

Por fim, uma medida que faz o balanceamento entre a precisão e o recall chamada F1 Score foi utilizada. Esta medida é descrita pela Equação 8.

$$F1Score = \frac{2*Precisão*Recall}{Precisão+Recall} \quad (8)$$

Para a primeira classe temos:

$$F1Score = \frac{2*1*0.5}{1+0.5} = 0.6667 \quad (9)$$

#### 4.4.2.5 Relatório de classificação

Uma ferramenta que facilita a extração destas medidas é o Relatório de Classificação que a biblioteca SCIKIT provê.

A utilização e os resultados do uso desta ferramenta podem ser observados na Figura 14.

Figura 14– Relatório de Classificação

```
print(classification_report(predicoes,y_test))
```

	precision	recall	f1-score	support
0	1.00	0.50	0.67	2
1	1.00	0.50	0.67	4
2	0.25	1.00	0.40	1
3	1.00	1.00	1.00	3
accuracy			0.70	10
macro avg	0.81	0.75	0.68	10
weighted avg	0.93	0.70	0.74	10

Fonte: Acervo do autor (2021)

Neste relatório, as métricas encontradas para cada classe são explicitadas. Pode-se notar que o algoritmo oferece uma identificação perfeita para o problema de desbalanceamento de eixo (Classe 3), uma performance razoável para o Regime Normal (Classe 0) e a falta de Tensão (Classe 1) porém oferece uma baixa performance para a detecção de Sobrecarga (Classe 2).

#### 4.4.3 Busca em grade

Após encontrar o modelo que entrega a melhor performance, pode-se usar uma técnica chamada “busca em grade” para estabelecer os melhores parâmetros para o algoritmo. Esta



técnica consiste em estabelecer diversos parâmetros que devem ser testados, verificar a sua saída e entregar os parâmetros que entreguem a melhor performance para o algoritmo.

O algoritmo que apresentou a melhor performance inicial veio da classe `DecisionTreeClassifier` do `SCIKIT`. Esta classe implementa um algoritmo de Árvore de Decisão.

Neste setup, diversos ajustes podem ser alterados como o tipo de critério utilizado, a profundidade máxima dentro outros como demonstrado na Figura 15.

Figura 15 – Busca em grade

```

criterion = ['gini', 'entropy']
max_depth = [2, 3, 4, 6, 8, 10, 12]
min_samples_split = [2, 3, 4, 6, 8, 10, 12]

parameters = {
    'criterion':criterion,
    'max_depth':max_depth,
    'min_samples_split':min_samples_split
}

dec_tree = DecisionTreeClassifier()
clf = GridSearchCV(dec_tree, parameters, scoring='f1_macro')

```

Fonte: Acervo do autor (2021)

## 4.5 INTEGRAÇÃO COM A FONTE GERADORA DE DADOS

Ao finalizar o trabalho de treino do algoritmo é necessário integrá-lo a fonte geradora de dados. Para tanto, é necessário exportar o modelo treinado e criar um algoritmo que, em tempo real, faz a leitura do microfone e entrega a predição feita pelo algoritmo.

### 4.5.1 Exportação do modelo

O primeiro passo para a construção do algoritmo em questão é a exportação do modelo criado nas etapas anteriores. Para isso, o módulo mais conveniente é o `PICKLE` pois o mesmo exporta um objeto `PYTHON` e salva o mesmo em um arquivo persistente no dispositivo.

A Figura 16 demonstra a sua utilização salvando o melhor classificador no diretório “Pre-ditores/classificador\_som\_industrial.pkl”.

Figura 16 – Exportação do modelo

```
import pickle as pickle

with open('Preditores/classificador_som_industrial.pkl', 'wb') as best_file:
    pickle.dump(best_clf, best_file)
```

Fonte: Acervo do autor (2021)

#### 4.5.2 Importação do modelo

Agora, já em outro arquivo, para importar este modelo gerado, basta usar o módulo PICKLE novamente como demonstrado a Figura 17.

Figura 17 – Importação do modelo

```
# Carregar o modelo gerado
path = 'Análise dos dados/Preditores/'
file_name = 'classificador.pkl'

print(os.path.isfile(path + file_name))

with open(path + file_name, 'rb') as file:
    clf = pickle.load(file)
```

Fonte: Acervo do autor (2021)

Com esta simples importação já é possível fazer previsões com o modelo carregado.

#### 4.5.3 Previsões com o modelo

Para fazer as previsões com o modelo basta usar o método “predict” do modelo carregado junto aos dados obtidos pela biblioteca PYAUDIO conforme a Figura 18 demonstra.

Figura 18 – Previsões com o modelo

```
data = stream.read(CHUNK) # Leitura do dados
dataInt = struct.unpack(str(CHUNK) + 'h', data) # Conversão de binários para inteiros
dataFFT = apply_fft(dataInt, CHUNK) # Aplicação da FFT

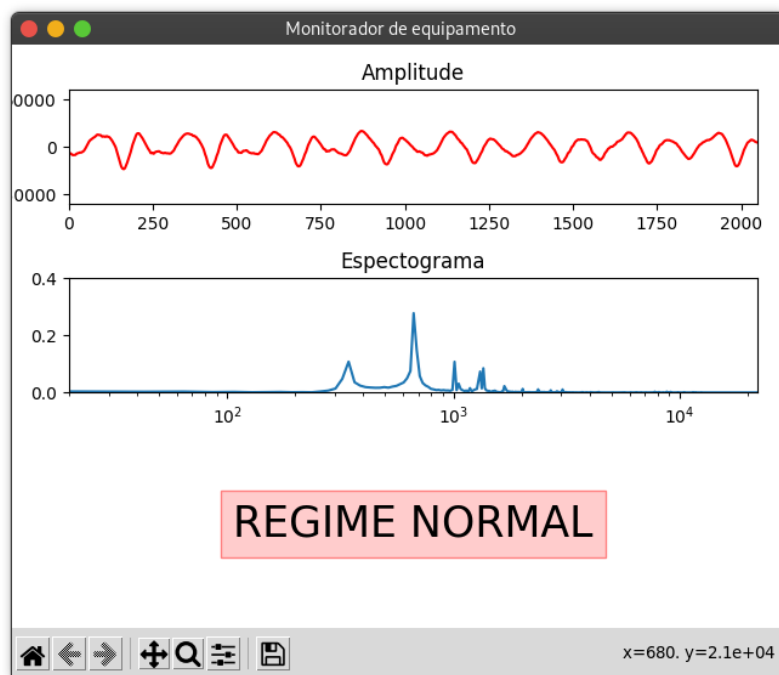
# Predição da saída
predict = clf.predict([dataFFT])
```

Fonte: Acervo do autor (2021)

#### 4.5.4 Interface gráfica e exibição dos resultados

Para facilitar a visualização dos dados e da saída predita, uma interface foi criada com a utilização da biblioteca MATPLOTLIB. Esta interface pode ser observada na Figura 19.

Figura 19 – Interface do sistema em tempo real

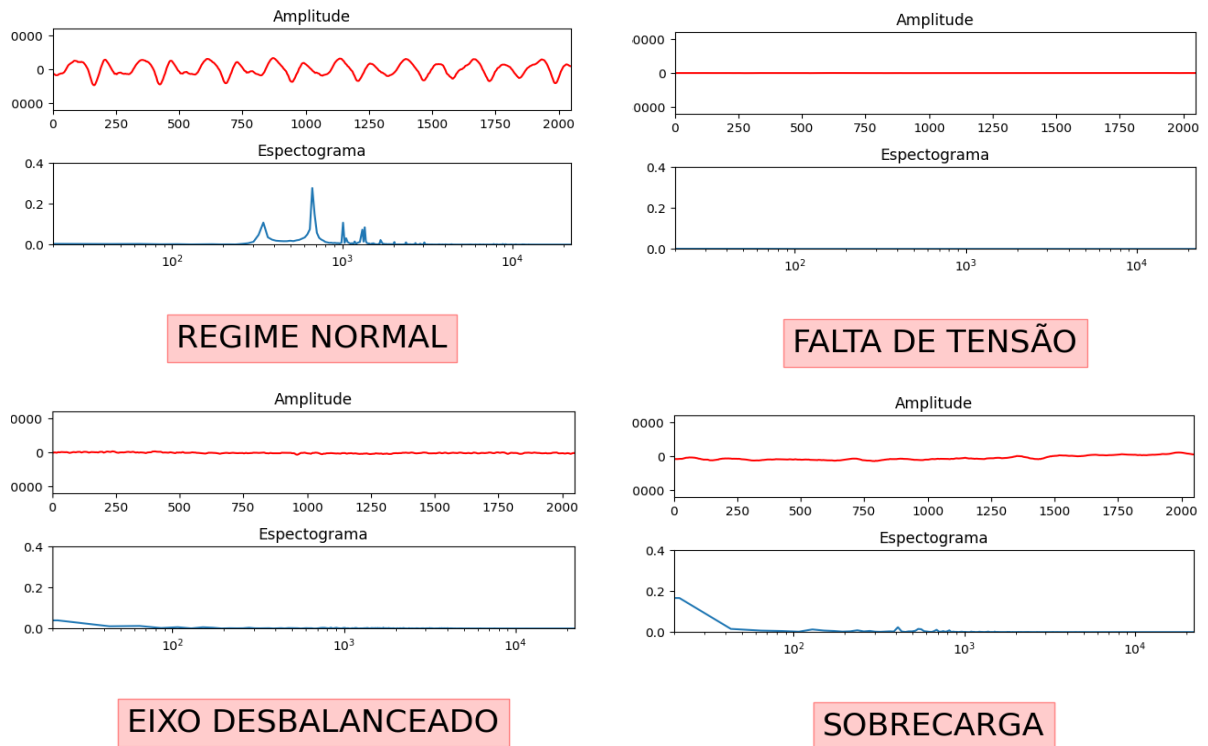


Fonte: Acervo do autor (2021)

O algoritmo completo pode ser visto no DETECTOR e no Apêndice B do presente trabalho.

Um demonstrativo com a interface funcionando para todos os defeitos explorados pode ser observado na Figura 20.

Figura 20 – Interface do sistema em tempo real, para cada defeito analisado



Fonte: Acervo do autor (2021)

## **5 CONSIDERAÇÕES FINAIS**

O objetivo inicial do presente trabalho era o projeto e implementação de um protótipo de sensor capaz de detectar falhas em equipamentos com a utilização de técnicas de aprendizado de máquina. O protótipo final se mostrou conciso e apresentou resultados aceitáveis.

Além do objetivo central, o foco aplicado ao desenvolvimento do projeto possibilita que este trabalho sirva como base geral para futuros projetos que integrem tecnologias como o aprendizado de máquina, exploração de dados e sensores em tempo real pois, de forma geral, ensina as bases de todo projeto de aprendizado de máquina bem como a forma em que as componentes de cada fase se encaixam.

Pode-se, desta forma, concluir que o presente trabalho foi um sucesso e atendeu os objetivos pelos quais foi inicialmente pensado.

## REFERÊNCIAS

OUR World Data. In: **Our World Data**. [S. l.], 1 abr. 2020. Disponível em: <https://our-worldindata.org/grapher/gdp-vs-happiness>. Acesso em: 25 jul. 2021.

TOLIYAT, Hamid; NANDI, Subhasis; CHOI, Seungdeog; MESHGIN-KELK, Homayoun. **Electric Machines: Modeling, Condition Monitoring, and Fault Diagnosis**. 1. ed. [S. l.]: CRC Press, 2017. 400 p. Disponível em: <https://www.amazon.com.br/Electric-Machines-Condition-Monitoring-Diagnosis-ebook/dp/B078JPPH7V>. Acesso em: 25 jul. 2021.

RAMALHO, Luciano. **Python Fluente: Programação Clara, Concisa e Eficaz**. 1. ed. [S. l.]: Novatec Editora, 2015. 800 p. Disponível em: <https://www.amazon.com.br/Python-Fluente-Programa%C3%A7%C3%A3o-Concisa-Eficaz/dp/857522462X>. Acesso em: 25 jul. 2021.

MCKINNEY, Wes. **Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython**. 2. ed. atual. [S. l.]: O'Reilly Media, 2017. 885 p. Disponível em: [https://www.amazon.com.br/Python-Data-Analysis-Wrangling-IPython-ebook/dp/B075X4LT6K/ref=sr\\_1\\_1?\\_\\_mk\\_pt\\_BR=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=Python+for+Data+Analysis&qid=1627221450&s=digital-text&sr=1-1](https://www.amazon.com.br/Python-Data-Analysis-Wrangling-IPython-ebook/dp/B075X4LT6K/ref=sr_1_1?__mk_pt_BR=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=Python+for+Data+Analysis&qid=1627221450&s=digital-text&sr=1-1). Acesso em: 25 jul. 2021.

AURELIEN, Geron. **Hands-On Machine Learning with Scikit-Learn and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems**. 2. ed. atual. [S. l.]: O'Reilly Media, 2017. 574 p. Disponível em: [https://www.amazon.com.br/Hands-Machine-Learning-Scikit-Learn-Tensor-Flow/dp/1491962291/ref=sr\\_1\\_7?\\_\\_mk\\_pt\\_BR=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=hands+on+machine+learning&qid=1627222031&sr=8-7&ufe=app\\_do%3Aamzn1.fos.25548f35-0de7-44b3-b28e-0f56f3f96147](https://www.amazon.com.br/Hands-Machine-Learning-Scikit-Learn-Tensor-Flow/dp/1491962291/ref=sr_1_7?__mk_pt_BR=%C3%85M%C3%85%C5%BD%C3%95%C3%91&dchild=1&keywords=hands+on+machine+learning&qid=1627222031&sr=8-7&ufe=app_do%3Aamzn1.fos.25548f35-0de7-44b3-b28e-0f56f3f96147). Acesso em: 25 jul. 2021.

UNPINGCO, Jose. **Python for Signal Processing: Featuring Ipython Notebooks**. 1. ed. [S. l.]: Springer, 2013. 128 p. Disponível em: <https://www.amazon.com.br/Python-Signal-Processing-Featuring-Notebooks/dp/3319013416>. Acesso em: 25 jul. 2021.

ZONST, Anders. **Understanding the FFT**. 2. ed. rev. [S. l.]: Citrus Pr, 2000. 180 p. Disponível em: <https://www.amazon.com/Understanding-Second-Revised-Anders-Zonst/dp/0964568152>. Acesso em: 25 jul. 2021.

CHOLLET, François. **Deep Learning with Python**. [S. l.]: Manning Publications, 2017. 384 p. Disponível em: <https://www.amazon.com.br/Deep-Learning-Python-Francois-Chollet/dp/1617294438>. Acesso em: 25 jul. 2021.

FRIZZO STEFENON, Stéfano; WALDRIGUES BRANCO, Nathielle; NIED, Ademir; WILDGRUBE BERTOL, Douglas; FINARDI, Erlon; SARTORI, Andreza; MEYER, L.H.; GREBOGI, Rafael. Analysis of training techniques of ANN for classification of insulators in electrical power systems. **Analysis of the Electrical Power System**, [S. l.], p. 1-14, 1 abr. 2020. Disponível em: [https://www.researchgate.net/publication/338886066\\_Analysis\\_of\\_training\\_techniques\\_of\\_ANN\\_for\\_classification\\_of\\_insulators\\_in\\_electrical\\_power\\_systems](https://www.researchgate.net/publication/338886066_Analysis_of_training_techniques_of_ANN_for_classification_of_insulators_in_electrical_power_systems). Acesso em: 25 jul. 2021.

VIB MASTER (Campo Bom/RS). Conheça as causas de falhas em motores elétricos. *In*: **Conheça as causas de falhas em motores elétricos**. [S. l.], 11 jun. 2018. Disponível em: <https://www.vibmaster.com.br/falhas-em-motores-eletricos/>. Acesso em: 25 jul. 2021.

SCHMIDT, Vinicio. Detecção de falhas em isoladores com Machine Learning. **Detecção de falhas em isoladores com Machine Learning**, [S. l.], p. 1, 1 abr. 2020. Disponível em: <https://vinicio-schmidt.medium.com/detecção-de-falhas-em-isoladores-com-machine-learning-659a9981f6fd>. Acesso em: 25 jul. 2021.

INDUSTRY. [S. l.: s. n.], 2014. Disponível em: <https://freesound.org/people/ikayuka/sounds/259501/>. Acesso em: 25 jul. 2021.

DETECTOR de falhas em máquinas por análise sonora. [S. l.], 2021. Disponível em: [https://github.com/vschmidt/machine\\_failure\\_detector](https://github.com/vschmidt/machine_failure_detector). Acesso em: 25 jul. 2021.

ALEXANDRE, Gonfalonieri. How to Build A Data Set For Your Machine Learning Project. **How to Build A Data Set For Your Machine Learning Project**, [S. l.], p. 1, 13 fev. 2019. Disponível em: <https://towardsdatascience.com/how-to-build-a-data-set-for-your-machine-learning-project-5b3b871881ac>. Acesso em: 25 jul. 2021.

BROWNLIE, Jason. How Much Training Data is Required for Machine Learning?. **Machine Learning Mastery**, [S. l.], p. 1, 24 jul. 2017. Disponível em: <https://machinelearningmastery.com/much-training-data-required-machine-learning/>. Acesso em: 25 jul. 2021.

HUMBY, Clive. Tech giants may be huge, but nothing matches big data: When Nasdaq stopped trading this week, it again showed how global firms are at the mercy of a power that created them. **The Guardian**, [S. l.], p. 1, 23 ago. 2013. Disponível em: <https://www.theguardian.com/technology/2013/aug/23/tech-giants-data>. Acesso em: 25 jul. 2021.

ENCICLOPÉDIA Britannica. *In*: **Machine Learning**. [S. l.], 2021. Disponível em: <https://www.britannica.com/technology/machine-learning>. Acesso em: 25 jul. 2021.

PYAUDIO. *In: PyAudio*. [S. l.], 25 jul. 2021. Disponível em: <https://people.csail.mit.edu/hubert/pyaudio/>. Acesso em: 25 jul. 2021.

PICKLE. *In: Pickle*. [S. l.], 25 jul. 2021. Disponível em: <https://docs.python.org/3.11/library/pickle.html>. Acesso em: 25 jul. 2021.

SCIKIT Learn. *In: Scikit Learn*. [S. l.], 25 jul. 2021. Disponível em: <https://scikit-learn.org/stable/>. Acesso em: 25 jul. 2021.

MATPLOTLIB. *In: Matplotlib*. [S. l.], 25 jul. 2021. Disponível em: <https://matplotlib.org/>. Acesso em: 25 jul. 2021.

JUPYTER. *In: Jupyter*. [S. l.], 25 jul. 2021. Disponível em: <https://jupyter.org/>. Acesso em: 25 jul. 2021.

PYTHON. *In: Python*. [S. l.], 25 jul. 2021. Disponível em: <https://www.python.org/>. Acesso em: 25 jul. 2021.



## APÊNDICE A – Algoritmo de gravação usado para obtenção dos dados

```

import os
from datetime import datetime
import pyaudio as pa
import wave

FORMAT = pa.paInt16
CHANNELS = 1
RATE = 44100 # Em Hz
CHUNK = 1024 * 2
RECORD_SECONDS = 10
BASE_PATH = os.path.dirname(os.path.realpath(__file__)) # Pasta base

OUTPUT_FOLDER_PATH = BASE_PATH + '/Saídas/1 - Som ambiente/' # Pasta
onde serão armazenados os áudios
TEST_TYPE = OUTPUT_FOLDER_PATH + '/0 - Regime Normal/' # Tipo de teste
que será executado
WAVE_OUTPUT_FILENAME = TEST_TYPE + str(datetime.now()) + '.wav' #
Nome do arquivo

# Criar uma instância do PyAudio
audio = pa.PyAudio()

stream = audio.open(format=FORMAT,
                    channels=CHANNELS,
                    rate=RATE,
                    input=True,
                    frames_per_buffer=CHUNK)

# Iniciando a gravação
print("Gravando...")

Recordframes = []

```

```
for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
    data = stream.read(CHUNK)
    Recordframes.append(data)

# Salvar o arquivo gravado
print("Fim da gravação, salvando arquivo...")

stream.stop_stream()
stream.close()
audio.terminate()

waveFile = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
waveFile.setnchannels(CHANNELS)
waveFile.setsampwidth(audio.get_sample_size(FORMAT))
waveFile.setframerate(RATE)
waveFile.writeframes(b''.join(Recordframes))
waveFile.close()
```

## APÊNDICE B – Algoritmo para coleta, predição e demonstração dos resultados em tempo real

```

import numpy as np
import os
import pickle as pkl
import pyaudio as pa
import struct
import matplotlib.pyplot as plt
from sklearn.svm import NuSVC

# Parâmetros
CHUNK = 1024 * 2
FORMAT = pa.paInt16
CHANNELS = 1
RATE = 44100 # Em Hz

# Carregar o modelo gerado
path = 'Análise dos dados/Preditores/'
file_name = 'classificador_som_industrial.pkl'

with open(path + file_name, 'rb') as file:
    clf = pkl.load(file)

# Funções auxiliares
def apply_fft(dataInt, CHUNK):
    """ Esta função retorna um array com a FFT """
    fft_value = np.fft.fft(dataInt)
    absolute_value = np.abs(fft_value)
    audio_freq_convert = absolute_value * 2 / (33000 * CHUNK)
    return audio_freq_convert

# Configurações do gravador
p = pa.PyAudio()

```

```

stream = p.open(
    format=FORMAT,
    channels=CHANNELS,
    rate=RATE,
    input=True,
    output=True,
    frames_per_buffer=CHUNK
)

data = stream.read(CHUNK)

# Converter os binários para inteiros
dataInt = struct.unpack(str(CHUNK) + 'h', data)

# Configurações da figura
fig, (ax1, ax2, ax3) = plt.subplots(3)

x = np.arange(0, 2*CHUNK, 2)
x_fft = np.linspace(0, RATE, CHUNK)

line, = ax1.plot(x, np.random.rand(CHUNK), 'r')
line_fft, = ax2.semilogx(x_fft, np.random.rand(CHUNK))

ax1.set_ylim(-60000, 60000)
ax1.set_xlim(0, CHUNK)

ax2.set_ylim(0, 0.4)
ax2.set_xlim(20, RATE/2)

ax3.text(0.5, 0.5, "CLASSE", size=25,
        ha='center', va='center',
        bbox=dict(boxstyle="square",
        ec=(1., 0.5, 0.5),

```

```

        fc=(1., 0.8, 0.8),
    )
)

ax3.axis('off')

fig.show()

# Loop de leitura
while True:
    data = stream.read(CHUNK) # Leitura do dados
    dataInt = struct.unpack(str(CHUNK) + 'h', data) # Conversão de binários para inteiros
    dataFFT = apply_fft(dataInt, CHUNK) # Aplicação da FFT

    # Predição da saída
    predict = clf.predict([dataFFT])

    # Encontrar a saída predita
    if(predict[0] == 0):
        ax3.texts[0].set_text('REGIME NORMAL')
    elif(predict[0] == 1):
        ax3.texts[0].set_text('FALTA DE TENSÃO')
    elif(predict[0] == 2):
        ax3.texts[0].set_text('SOBRECARGA')
    else:
        ax3.texts[0].set_text('EIXO DESBALANCEADO')

    # Exposição dos dados
    line.set_ydata(dataInt)
    line_fft.set_ydata(np.abs(np.fft.fft(dataInt))*2/(33000*CHUNK))

    fig.canvas.draw()
    fig.canvas.flush_events()

```

