UNIVERSIDADE FEDERAL DE SÃO CARLOS - UFSCar
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA - CCET
DEPARTAMENTO DE ENGENHARIA MECÂNICA - DEMec

Rafael Luccas Martins de Carvalho

USO DE INTELIGÊNCIA ARTIFICIAL PARA IDENTIFICAÇÃO DE CONDIÇÕES DE USINAGEM A PARTIR DE SINAIS DE ÁUDIO

São Carlos - SP
2025

Rafael Luccas Martins de Carvalho

USO DE INTELIGÊNCIA ARTIFICIAL PARA IDENTIFICAÇÃO DE CONDIÇÕES DE USINAGEM A PARTIR DE SINAIS DE ÁUDIO

Trabalho de conclusão de curso apresentado ao curso de graduação em Engenharia Mecânica da Universidade Federal de São Carlos, para obtenção do título de bacharel em Engenharia Mecânica.
Orientador: Prof. Dr. Sidney Bruce Shiki

São Carlos - SP

2025

FUNDAÇÃO UNIVERSIDADE FEDERAL DE SÃO CARLOS

COORDENAÇÃO DO CURSO DE ENGENHARIA MECÂNICA - CCEMec/CCET

Rod. Washington Luís km 235 - SP-310, s/n - Bairro Monjolinho, São Carlos/SP, CEP 13565-905
Telefone: (16) 33519703 - http://www.ufscar.br

DP-TCC-FA nº 32/2025/CCEMec/CCET

Graduação: Defesa Pública de Trabalho de Conclusão de Curso

Folha Aprovação (GDP-TCC-FA)

FOLHA DE APROVAÇÃO

SIDNEY BRUCE SHIKI

USO DE INTELIGÊNCIA ARTIFICIAL PARA IDENTIFICAÇÃO DE CONDIÇÕES DE USINAGEM A PARTIR DE SINAIS DE ÁUDIO

Trabalho de Conclusão de Curso

Universidade Federal de São Carlos – Campus São Carlos

São Carlos, 25 de julho de 2025

ASSINATURAS E CIÊNCIAS

Cargo/Função	Nome Completo
Orientador	Sidney Bruce Shiki
Membro da Banca 1	Armando Ítalo Sette Antonialli
Membro da Banca 2	Alvaro Araujo de Moraes

Documento assinado eletronicamente por **Sidney Bruce Shiki**, **Docente**, em 25/07/2025, às 11:30, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Armando Italo Sette Antonialli**, **Docente**, em 25/07/2025, às 11:53, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufscar.br/autenticacao, informando o código verificador **1926631** e o código CRC **563604EC**.

Referência: Caso responda a este documento, indicar expressamente o Processo nº 23112.023424/2025-47

SEI nº 1926631

Modelo de Documento: Grad: Defesa TCC: Folha Aprovação, versão de 02/Agosto/2019

Documento assinado digitalmente
ALVARO ARAUJO DE MORAES
Data: 25/07/2025 15:26:13-0300
Verifique em https://validar.iti.gov.br

Dedico este trabalho à minha família, fonte de apoio em toda a minha jornada.

Agradecimentos

Agradeço à Deus pela vida e cuidado sempre presente.

À minha família, pelo apoio e carinho contumazes.

Ao meu orientador neste trabalho, Sidney Bruce que, capaz e pacientemente dirigiu meus esforços até esta obra final.

"Criança, estou lhe contando a sua história, não a dela. A ninguém será contada a história de outro."

C.S. Lewis - As Crônicas de Nárnia.

Resumo

As operações de usinagem são de grande importância para a indústria mundial, de forma que a qualidade e eficiência das máquinas que realizam estes processos são fundamentais para para os resultados de uma indústria. O monitoramento destas máquinas ganham assim relevância ao impedir falhas e garantir a qualidade de processos. Embora a maioria dos sensores utilizados para monitoramento sejam acelerômetros, e as análises no campo vibracional, o uso de microfones surge como uma possibilidade poderosa e de baixo custo, associada ao uso de algoritmos de inteligência artificial (IA) para analisar o sinal sonoro. Este trabalho utiliza técnicas de IA para classificar sinais de áudio provenientes de tornos de controle numérico computadorizado (CNC) durante processo de torneamento de um cilindro de liga de titânio (Ti-6Al-4V ELI) , capturados a partir de um *smartphone*. O sinal sonoro obtido de 9 testes, com parâmetros de usinagem diferentes, e 2 arquivos de ruído de fundo, foi processado e os modelos Floresta Aleatória, Regressão Logística, k-Vizinhos Mais Próximos (kNN na sigla em inglês) e Máquina de Vetores de Suporte (SVM na sigla em inglês) foram testados para se identificar o mais capaz de predizer os testes e, consequentemente os parâmetros de usinagem de cada sinal. Todos os modelos tiveram bom desempenho, com área sob a curva característica de operação do receptor (AUC) > 93%. Entretanto o modelo de regressão logística se destacou com 96,2% de Acurácia e 99,75% AUC. Estes resultados mostram grande potencial para utilização dessa técnica de monitoramento.

Palavras-chave: Monitoramento de processos; torneamento; análise sonora; aprendizado de máquina; usinagem CNC.

Abstract

Machining operations are crucial to global industry, so the quality and efficiency of the machines that perform these processes are crucial to a company's bottom line. Monitoring these machines thus gains relevance by preventing failures and ensuring process quality. Although most sensors used for monitoring are accelerometers, and vibration analysis is essential, the use of microphones emerges as a powerful and low-cost option, coupled with the use of artificial intelligence (AI) algorithms to analyze sound signals. This work uses AI techniques to classify audio signals from computer numerical controlled (CNC) lathes during the turning process of a titanium alloy cylinder (Ti-6Al-4V ELI), captured from a smartphone. The sound signal obtained from nine tests with different machining parameters and two background noise files was processed, and the Random Forest, Logistic Regression, k-nearest neighbours (kNN), and support vector machine (SVM) models were tested to identify the most capable of predicting the tests and, consequently, the machining parameters of each signal. All models performed well, with area under the receiver-operating curve (AUC) > 93%. However, the logistic regression model stood out with 96.2% accuracy and 99.75% AUC. These results demonstrate great potential for this monitoring technique.

Keywords: Process monitoring; turning; sound analysis; machine learning; CNC machining.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

Acc	Acurácia
a_p	Profundidade de corte
AUC	Área sob a curva ROC (*Area Under Curve*)
CNC	Comando Numérico Computadorizado
DFT	Transformada Discreta de Fourier *(Discrete Fourier Transform*)
DTFT	Transformada de Fourier de Tempo Discreto (*Discrete Time Fourier Transform*)
Err	Erro
f	Avanço
F1	Média harmônica balanceada entre *recall* e precisão
FFT	Transformada Rápida de Fourier *(Fast Fourier Transform)*
FN	Falso Negativo (*False Negative*)
FP	Falso Positivo (*False Positive*)
IA	Inteligência Artificial
kNN	k-Vizinho Mais Próximo (*k-Nearest Neighbor*)
n	Velocidade de rotação da peça
ROC	Curva Característica de Operação do Receptor (*Receiver-operating curve*)
RPM	Rotações por minuto
SVM	Máquina de Vetores de Suporte (*Support Vector Machine*)
TN	Verdadeiro Negativo (*True Negative*)
TP	Verdadeiro Positivo (*True Positive*)
v_c	Velocidade de corte
v_f	Velocidade de avanço

Declaração de Uso de Inteligência Artificial

Eu, Rafael Luccas Martins de Carvalho, o autor deste manuscrito, declaro que:

1. **Uso de Ferramentas de Inteligência Artificial:**
   Durante o desenvolvimento deste trabalho, as seguintes ferramentas de inteligência artificial foram utilizadas:

Ferramenta(s): Copilot, Google Gemini

Propósito(s):

- ☑ **Revisão de linguagem:** melhoria da gramática, ortografia, clareza e estilo do texto.
- ☐ **Tradução**: conversão do manuscrito ou partes dele para outro idioma.
- ☑ **Geração de texto**: criação de rascunhos, sugestões ou partes específicas do conteúdo, devidamente revisadas pelos autores.
- ☐ **Análise de dados:** processamento e interpretação inicial de conjuntos de dados.
- ☐ **Visualização de dados ou imagens**: auxílio na geração ou refinamento de figuras, gráficos ou ilustrações.
- ☐ **Sugestões metodológicas**: fornecimento de recomendações ou ideias para aprimorar abordagens de pesquisa.
- ☐ **Outros (especificar)**:

2. **Autoria e Supervisão Humana:**

A utilização de ferramentas de inteligência artificial foi exclusivamente para os propósitos acima declarados. Toda supervisão, validação e autoria intelectual do manuscrito são de responsabilidade do autor humano.

3. **Conformidade com Princípios Éticos:**

O uso das ferramentas de IA seguiu práticas éticas e não incluiu atividades como fabricação de dados, plágio, manipulação de figuras ou qualquer prática que comprometa a integridade científica.

4. **Transparência**:

Declaramos que as informações acima são completas e verdadeiras. Caso seja identificada qualquer inconsistência ou uso inadequado, aceitamos que o manuscrito seja submetido a investigação e, se necessário, rejeitado ou retratado.

Sumário

1. Introdução

As operações de usinagem são aquelas que, ao conferir forma, dimensões ou acabamento a uma peça, removem uma porção do material (chamado cavaco) desta peça com o auxílio de uma ferramenta (Ferraresi, 1970). Essas operações estão presentes quase totalidade da produção industrial mundial, representando cerca de 26% do produto interno bruto mundial ("Share of economic sectors in the global gross domestic product (GDP) from 2014 to 2024", 2025). Apenas nos EUA, anualmente a indústria consome mais de $100 bi de dólares em processos de remoção de metal, pois a vasta maioria dos produtos manufaturados requerem usinagem em alguma etapa do processo de fabricação (Black; Kohser, 2008).

Mohanty (2015) explica que para alcançar altos níveis de produção industrial, é essencial que as máquinas industriais estejam funcionando dentro das suas especificações de projeto e capacidade instalada. A manutenção de máquinas depende do tipo de máquina, da gravidade dos defeitos nas máquinas e das consequências que eles podem ter na operação geral da planta. Equipamentos em funcionamento parcial ou totalmente parados podem gerar gargalos numa linha de produção, atrasos em entregas, problemas logísticos e, consequentemente, elevadas perdas financeiras. A manutenção correta e no momento correto é uma busca constante da engenharia a fim de maximizar a eficiência produtiva de uma planta industrial.

Durante a operação, as máquinas emitem informações ou sinais na forma de ruído, vibração, temperatura, condição do óleo lubrificante, qualidade e quantidade da corrente do motor consumida, e similares. Esses sinais da máquina são adquiridos pela instalação de transdutores para medir os parâmetros mecânicos da máquina. Os sinais assim obtidos são geralmente analógicos e existem o tempo todo. Para criar informações significativas a partir desses sinais, os sinais são convertidos para o domínio digital por conversores analógico-digitais. Os dados digitais discretos correspondentes ao sinal analógico assim obtido são analisados em computadores. Há softwares disponíveis para armazenar e manipular eficientemente os grandes dados digitais coletados das máquinas. Esses dados podem ser usados em algoritmos desenvolvidos para detecção de falhas em máquinas. Uma vez diagnosticada a falha nas máquinas, medidas corretivas podem ser iniciadas para que a máquina tenha uma longa vida útil e a planta tenha alta produtividade (Mohanty, 2015).

Os avanços tecnológicos permitem a criação de métodos cada vez mais inteligentes, eficientes e a custos menores para a obtenção e análise dos sinais gerados pelas máquinas de produção. O conceito de Internet Industrial, por exemplo, como definido por Gilchrist (2016), traz a ideia de que a Internet Industrial proporciona uma maneira de obter melhor visibilidade e insights sobre as operações e ativos da empresa por meio da integração de sensores de máquina, software e sistemas de computação e armazenamento em nuvem de *back-end*. Utilizando como *feedback* os resultados obtidos com a análise de grandes conjuntos de dados por meio de análises avançadas, a Internet Industrial pode fornecer um método para transformar os processos operacionais. Os ganhos de negócios são alcançados por meio de ganhos de eficiência operacional e produtividade acelerada, o que resulta na redução de paradas não planejadas e na otimização da eficiência e, consequentemente, dos lucros.

Em paralelo com o avanço em conectividade e análise de grandes volumes de dados, está o recente avanço das técnicas de aprendizado de máquina ou inteligência artificial (IA). Aprendizado de máquina pode ser utilizado para prever falhas, otimizar processos e melhorar a qualidade da produção. Perdigão et al. (2021) discutem o papel da IA na otimização de processos industriais, destacando como a análise preditiva pode reduzir custos e aumentar a eficiência.

Da Costa (2024) apresenta que ferramentas com IA podem ser capazes de detectar, e em alguns casos até corrigir, falhas, anomalias ou desvios que, sem essas ferramentas, seriam muito mais dificilmente ou tardiamente identificados. O monitoramento mais frequente e com diagnósticos mais precisos, rápidos e eficientes, permitidos pelo uso de modelos de IA, reduzem os custos de manutenção, retrabalho e o impacto negativo de falhas em processos industriais. Dessa forma, o uso de IA é capaz também de aprimorar operações industriais, a qualidade dos produtos, a segurança industrial (inclusive melhorando condições de trabalho e prevenindo acidentes) e a sustentabilidade de operações industriais.

Em qualquer processo industrial, calor e vibração (muitas vezes audível) são gerados, sendo na maioria das vezes, processos indesejados que geram perda de eficiência. Entretanto, o sinal destes “subprodutos” pode ser utilizado para obter informações sobre o funcionamento da máquina e sua necessidade de manutenção. Quando uma máquina apresenta um defeito durante a operação, o resultado geralmente é ruído e vibração excessivos, mas ao realizar uma análise detalhada do sinal de vibração da máquina, o defeito e sua causa podem ser determinados (Mohanty, 2015).

Embora haja diversidade de pesquisas em aplicações de acelerômetros e lasers para medir e analisar as vibrações de uma máquina, poucos trabalhos investigam possibilidades utilizando sensores sonoros.

1.1 Objetivo geral

O objetivo principal deste trabalho é aplicar técnicas de aprendizado de máquina visando a classificação de sinais de áudio provenientes de tornos CNC durante processo de torneamento capturados a partir de dispositivos móveis.

1.2 Objetivos específicos

- Processar os arquivos de áudio para permitir o desenvolvimento de modelos de aprendizado de máquina;
- Desenvolver e treinar modelos de aprendizado de máquina;
- Analisar e validar os modelos de aprendizado de máquina;
- Definir melhorias para próximas etapas da pesquisa.

2. Fundamentação teórica

2.1 Parâmetros de usinagem

O resultado do processo de usinagem de metais é influenciado por uma série de parâmetros, tais como qual o processo em execução (torneamento, fresamento, aplainamento, furação, etc) potência da máquina, material a ser usinado e geometria da ferramenta. Entretanto, dentro de um mesmo processo, máquina e materiais, é possível variar alguns parâmetros dinâmicos que podem modificar completamente o resultado obtido. Ferraresi, (2014) define tais parâmetros, dos quais se destacam:

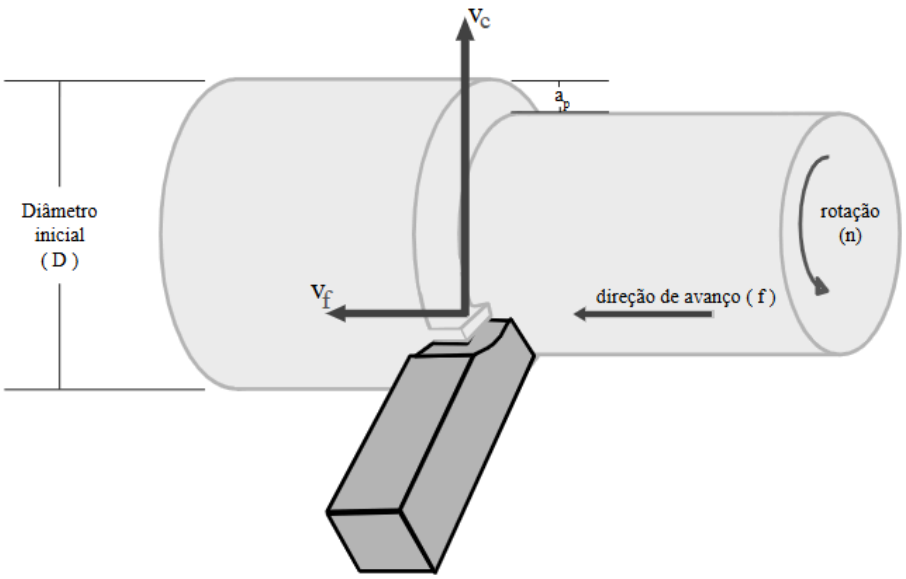
- **Avanço (f)**: é o percurso de avanço em cada volta ou em cada curso da peça ou da ferramenta, tipicamente medida em milímetros por volta.
- **Profundidade de corte (a_p)**: é a profundidade ou largura de penetração da aresta principal de corte, medida numa direção perpendicular ao plano de trabalho, medida em milímetros.
- **Velocidade de rotação da peça (n)**: velocidade de rotação da peça ou da ferramenta da corte, tipicamente medida em rotações por minuto (RPM).

A partir destes parâmetros, conhecido o diâmetro inicial da peça, pode-se calcular diversas métricas dependentes, tais como:

- **Velocidade de corte (v_c)**: a velocidade instantânea do ponto de referência da aresta cortante, segundo a direção e sentido de corte. Tipicamente medida em metros por minuto e calculada por $v_c = \frac{\pi.D.n}{1000}$, sendo D o diâmetro inicial da peça, medido em milímetros.
- **Velocidade de avanço (v_f)** : é a velocidade instantânea da ferramenta, segundo a direção e sentido de avanço. Tipicamente medida em milímetros por minuto e calculada por $v_f = f.n$

A figura 1 traz uma representação gráfica dos parâmetros e métricas do processo de torneamento, apresentados anteriormente.

Figura 1: Parâmetros de usinagem no processo de torneamento.



Fonte: Próprio autor.

A variação dessas métricas , dentre outras, impactam intensamente outros parâmetros de valor econômico e de engenharia, tais como potência necessária, tempo de corte, desgaste da ferramenta de corte, vida útil da ferramenta de corte, rugosidade final da peça, tolerância geométrica, formato do cavaco, custo de produção, entre outros. Dessa forma, o monitoramento eficiente dos parâmetros de corte permite maior controle sobre o processo de usinagem e fortalecem a qualidade e a economia da produção (Machado, 2015).

2.2 Instrumentação de máquinas

Os sinais das máquinas precisam ser medidos e analisados para interpretar suas condições. Para isso, a máquina deve primeiro ser instrumentada com o transdutor apropriado para medir o parâmetro mecânico que se manifesta como um sinal da máquina. Mohanty (2015) afirma que, na maioria das máquinas em todo o mundo, as suas condições são monitoradas por meio de medições de vibração, sendo que, na maioria dos casos, as vibrações são medidas usando acelerômetros piezoelétricos de contato.

Devido aos avanços na eletrônica, os sistemas tornaram-se menores e mais leves e portáteis. A bateria como fonte de alimentação para o sistema de instrumentação também evoluiu consideravelmente para uma relação potência-peso mais alta, como células de íons de

lítio. Os sistemas de aquisição de dados tornaram-se muito portáteis e geralmente são sistemas baseados em computadores pessoais com uma interface USB (*Universal Serial Bus*) do tipo *plug-and-play*. Existem algoritmos de processamento digital de sinais que são rápidos e podem até mesmo determinar as características de um sinal adquirido de uma máquina rotativa com duração muito curta (Mohanty, 2015).

Nesse sentido, estudos têm sido conduzidos para reduzir custos e ampliar a gama de equipamentos capazes de monitorar condições de corte e parâmetros de usinagem. Porto (2023), analisa a capacidade de um microfone de celular, equipamento mais simples e acessível do que linhas "profissionais", de monitorar um processo de torneamento de liga de titânio. Ao comparar e analisar os dados dos arquivos de áudio com os dados obtidos por um acelerômetro, Porto conclui que, tanto as gravações de áudio, quanto o acelerômetro, geraram dados similares, capazes de permitir o monitoramento confiável de processos de usinagem, refletindo o estado da ferramenta de corte.

2.3 Acústica e análise de sinais

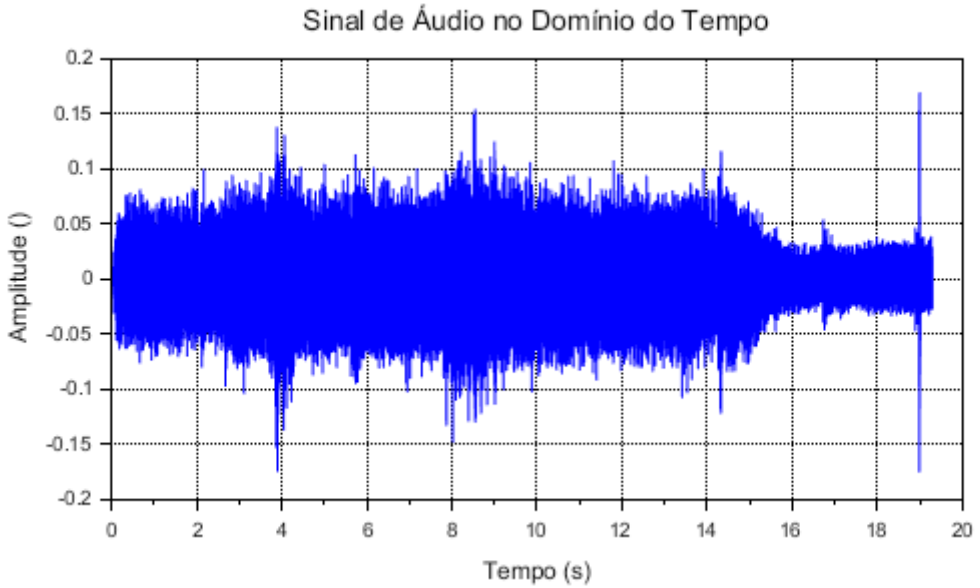
Mohanty (2015) apresenta que a acústica é o estudo da geração, propagação e recepção do som que é ouvido por um ser humano, resultado das ondas de pressão sonora incidentes no tímpano humano. Essas ondas sonoras requerem um meio elástico para sua propagação, e a natureza dessas ondas pode ser longitudinal, transversal, de flexão, cisalhamento ou de superfície. No ar, as ondas sonoras que se propagam são longitudinais por natureza e em sólidos elas geralmente são de cisalhamento ou flexão. A velocidade de propagação da frente de onda depende da natureza da onda e das propriedades do meio.

A amplitude da onda de pressão que chega ao tímpano proporciona a sensação de audição, sendo que os seres humanos conseguem ouvir ondas sonoras apenas em uma faixa de frequência específica, conhecida como faixa de frequência audível de 20 Hz a 20 kHz. Qualquer onda sonora com frequência abaixo de 20 kHz é conhecida como infrassônica, e ondas sonoras com frequência acima de 20 kHz são conhecidas como ultrassônicas (Mohanty, 2015).

Uma das ferramentas mais úteis para analisar sinais de áudio é a transformada de Fourier. Este método criado por Jean-Baptiste Joseph Fourier (1768-1830), transforma o domínio de uma função do tempo para a frequência (Heideman et al, 1985). Isso permite identificar as frequências com maiores amplitudes, em cada janela temporal analisada, e as

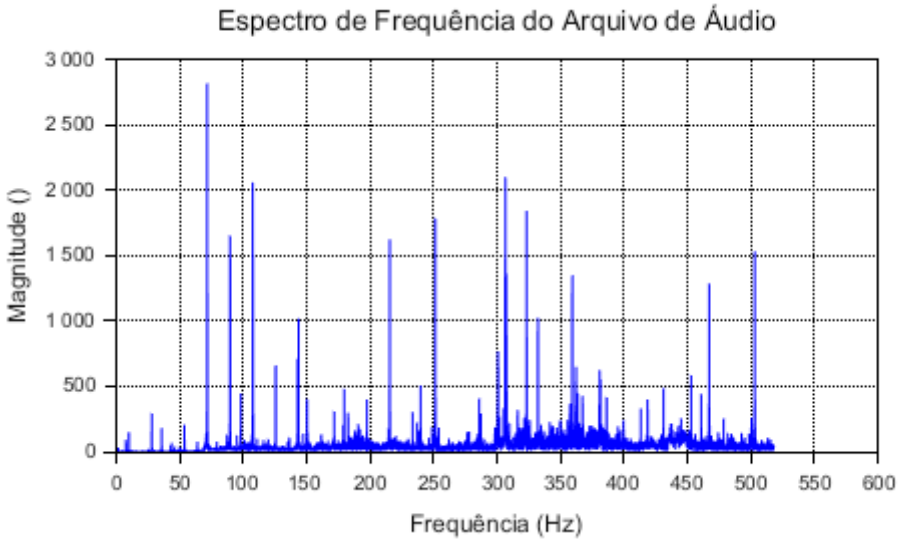
características únicas e distintivas do sinal analisado. As figuras 2 e 3 apresentam, respectivamente, um sinal de áudio no domínio do tempo e no domínio da frequência.

Figura 2: Sinal de áudio no domínio do tempo



Fonte: Próprio Autor.

Figura 3 : Sinal de áudio no domínio da frequência.

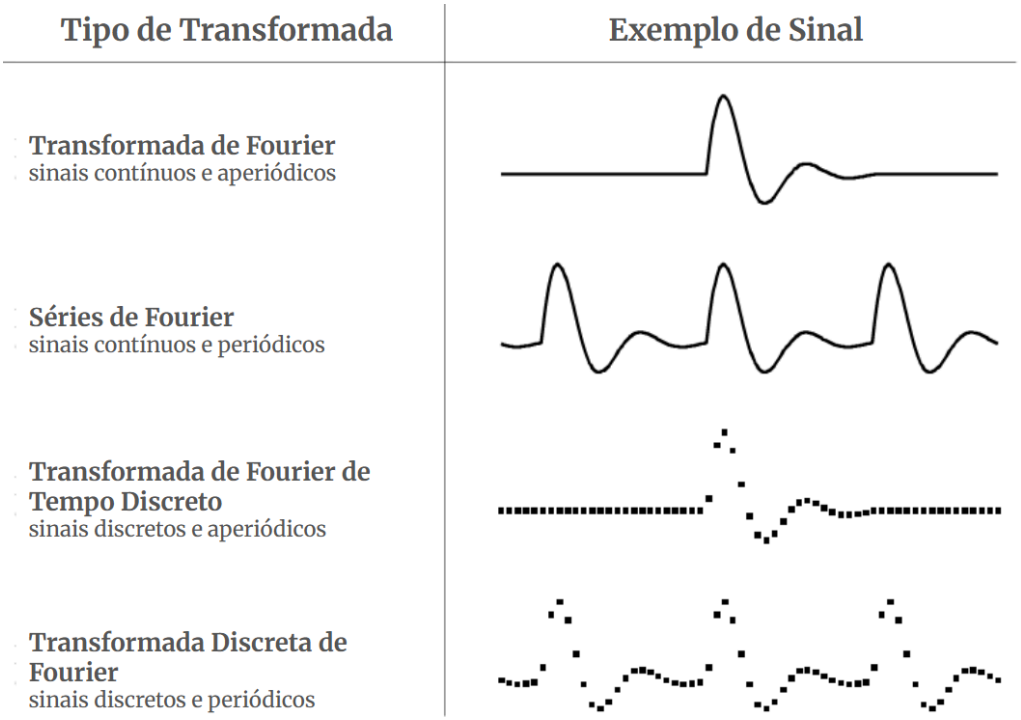


Fonte: Próprio Autor.

Smith (1999) mostra que a transformada de Fourier pode ser aplicada aos quatro tipos possíveis de sinais: funções contínuas ou discretas, periódicas ou não periódicas (também chamadas aperiódicas). Quando aplicadas a sinais contínuos e aperiódicos, a transformação é

chamada apenas de Transformada de Fourier. Quando aplicada a sinais contínuos e periódicos, como ondas senoidais e quadradas, a transformação é chamada de Séries de Fourier. Para sinais discretos e aperiódicos, a transformada é chamada de Transformada de Fourier de Tempo Discreto (*Discrete Time Fourier Transform*, DTFT). Quando aplicada a sinais periódicos e discretos, ela é chamada de Transformada Discreta de Fourier (*Discrete Fourier Transform*, DFT). A figura 4 ilustra os quatro tipos de transformada de Fourier, como descrito.

Figura 4: Ilustração dos quatro tipos possíveis de Transformada de Fourier.



Fonte: Adaptada de Smith (1999)

Esses quatro tipos de sinais são definidos num domínio de infinito negativo até infinito positivo e infinitas senoides são necessárias para sintetizar um sinal aperiódico. Dessa forma, para se calcular sinais finitos e não periódicos, define-se que o sinal se expande de forma infinita, mas com valores nulos, e a função é repetida para ser considerada periódica. Por isso, em cálculos com computadores digitais, só é possível realizar DFTs.

O cálculo de DFT era bastante complexo até que, em 1965, J. W. Cooley e J. W. Tukey publicaram um algoritmo capaz de reduzir a complexidade da transformação, chamando esse processo de Transformada Rápida de Fourier (*Fast Fourier Transform*, FFT). Embora diversos pesquisadores, como Carl Friedrich Gauss, tivessem trabalhado em

algoritmos semelhantes, a FFT de Cooley-Tokey tornou-se rapidamente bastante popular e revolucionou o processamento de sinais digitais (Heideman et al, 1985).

2.4 Desenvolvimento da Inteligência Artificial

O termo inteligência artificial (IA) foi cunhado em 1956 por John McCarthy, que a definiu como a "ciência e engenharia de construir e tornar máquinas inteligentes" (VALDATI, 2020). George F. Luger (2013) observa que, embora o nome seja moderno, a busca por uma "mente racional" em tecnologia tem raízes na filosofia humana, evoluindo com o avanço da computação.

O desenvolvimento da IA moderna teve altos e baixos, com períodos de euforia e de descrença, conhecidos como "invernos da IA". O progresso pode ser resumido por marcos históricos (MUTHUKRISHNAN, 2020):

- **1943:** Warren S. McCulloch e Walter Pitts criam um modelo de neurônio artificial para aprendizado.
- **1950:** Alan Turing propõe o "teste de Turing" para avaliar a inteligência de uma máquina.
- **1956:** A Conferência de Dartmouth define a IA como campo de estudo.
- **1958:** Frank Rosenblatt desenvolve o "perceptron", um modelo fundamental para redes neurais.
- **1968-1973:** Após decepções, um estudo de James Lighthill (1973) destaca o pouco progresso, levando ao primeiro "inverno da IA".
- **1985-1986:** David E. Rumelhart, Geoffrey E. Hinton e Ronald J. Williams introduzem a "retropropagação", revolucionando o aprendizado de redes neurais.
- **Início de 1990:** A falta de poder computacional leva ao segundo "inverno da IA".
- **Fim de 1990:** O ressurgimento acontece com o aumento do poder de processamento. Em 1997, o computador Deep Blue da IBM vence o campeão mundial de xadrez, aumentando o interesse na área.
- **2006 e 2012:** O acesso a grandes volumes de dados e o avanço das GPUs permitem o desenvolvimento do *deep learning* e o avanço no processamento de fala e imagens.

A IA é um campo de estudo multidisciplinar (VALDATI, 2020) com aplicações em diversas áreas. Existem três tipos de IA (LUDERMIR, 2021):

- **IA Focada (ou Fraca):** Desenvolvida para resolver problemas específicos, como é o caso das IAs atuais.
- **IA Generalizada (ou Forte):** Capaz de realizar diversas tarefas de forma comparável a um humano, o que ainda não foi totalmente alcançado.
- **IA Superinteligente:** Uma IA significativamente mais capaz que os seres humanos, um nível ainda não alcançado.

2.5 Modelos de aprendizado de máquina

O aprendizado de máquina pode ser dividido entre supervisionado e não supervisionado. No caso não supervisionado, o objetivo do algoritmo é determinar se é possível agrupar de alguma maneira os exemplos fornecidos, formando agrupamentos, chamados também de *clusters*. No aprendizado supervisionado, é apresentado para o algoritmo de aprendizado um conjunto de exemplos de treinamento, tipicamente um vetor de valores de características, para os quais é conhecido o rótulo da classe associada. Dessa forma, o algoritmo supervisionado tem por objetivo construir um classificador capaz de determinar corretamente a classe de novos exemplos ainda não rotulados. Se as opções de rótulos de classe forem valores discretos, trata-se de um problema de classificação. Caso sejam valores contínuos, tem-se um problema de regressão (Rezende, 2005).

No presente trabalho, todos os efeitos das mudanças de parâmetros e condições de usinagem são classificados unificadamente como efeito de variações de testes. Dessa forma, ao invés de se trabalhar com diversas variáveis contínuas, pode-se desenvolver o algoritmo para uma única variável discreta, tornando este um problema de classificação.

Salih et al (2021) trazem que o processo de classificação de dados passa por dois estágios: treinamento e teste. Durante o estágio de treinamento e aprendizagem, um classificador é aprendido como alvo, enquanto durante o segundo estágio, a fase de teste, o modelo construído é usado para prever os rótulos de classe para um determinado dado. Antes de aplicar os classificadores, o pré-processamento dos dados ajuda o modelo de classificação a diminuir o tempo e a complexidade, removendo dados irrelevantes para melhorar a eficiência dos algoritmos do classificador.

Os algoritmos de classificação mais populares são abaixo listados e explicados nas seções seguintes do presente texto:

- Floresta Aleatória (*Random Forest*)
- Regressão Logística (*Logistic Regression*)
- Máquina de Vetores de Suporte (*Support Vector Machine*, SVM)
- k-Vizinho Mais Próximo (*k-Nearest Neighbor*, k-NN)

2.5.1 Floresta Aleatória (*Random Forest*)

Rezende et. al (2005) define uma árvore de decisão como uma estrutura de dados que, através de um conjunto de regras capazes de classificar um conjunto de dados. A árvore de decisão é definida recursivamente como:

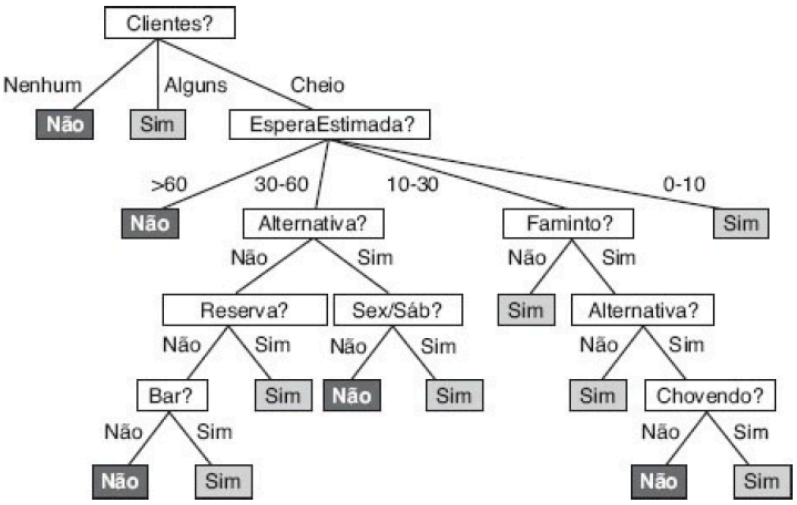
- um nó folha, que corresponde a uma classe (a resposta)
- um nó de decisão que contém um teste sobre algum atributo. Para cada resultado do teste existe uma aresta para uma subárvore e cada subárvore tem a mesma estrutura da árvore.

Russel e Norvig (2004) apresentam como exemplo uma árvore de decisão para decidir a espera ou não de uma mesa num restaurante. Inicialmente, lista-se os atributos a serem considerados, então define-se parâmetros para as decisões sobre cada atributo.

1. **Alternativa**: Se há um restaurante alternativo apropriado por perto.
2. **Bar**: Se o restaurante tem uma área de bar confortável onde se pode esperar.
3. **Sex/Sáb**: Verdadeiro às sextas e sábados.
4. **Faminto**: Se estamos com fome.
5. **Clientes**: Quantas pessoas estão no restaurante (os valores são: Nenhum, Alguns e Cheio).
6. **Chovendo**: Se está chovendo do lado de fora.
7. **Reserva**: Se fizemos uma reserva.
8. **Espera Estimada**: A espera estimada pelo gerente (0-10 minutos, 10-30, 30-60, >60).

Por fim, define-se, para cada atributo e parâmetro, qual decisão será tomada. A figura 5 apresenta essa árvore de decisão exemplificada.

Figura 5: Exemplo de árvore de decisão: definir se esperar por uma mesa num restaurante.



Fonte: Russel e Norvig (2004)

Marsland (2009) descreve que o modelo de floresta aleatória se baseia na criação de uma grande quantidade de árvores de decisão, formando uma floresta. Cada uma dessas árvores de decisão são modelos especialistas separados, independentes e treinados com subconjuntos de dados ligeiramente diferentes dos demais. Além disso, para aumentar a aleatoriedade e diversidade dentro do modelo, em cada nó de cada árvore, que é quando a árvore toma uma decisão, o Random Forest não considera todas as características disponíveis nos dados (*features*), dessa forma o algoritmo evita que características muito frequentes controlem toda a árvore, aumentando a diversidade das árvores. O efeito dessas duas formas de aleatoriedade é reduzir a variância sem afetar o viés. Outro benefício disso é que não há necessidade de podar as árvores.

Uma vez treinado o modelo, para definir a classificação de um conjunto de dados, cada árvore recebe e classifica esses dados independentemente, "votando" numa classificação. Então o modelo define a classificação como a maior número de votos como a classificação final.

O modelo tem dois parâmetros principais:

- **Número de árvores**: quantas serão utilizadas.
- **Mínimo de amostras** consideradas no nó folha: selecionado para evitar que o modelo se ajuste demais aos dados de treinamento (*overfitting*).

2.5.2 Regressão Logística (*Logistic Regression*)

Fávero (2017) apresenta que as técnicas de regressão logística são utilizadas quando o fenômeno a ser estudado é avaliado de forma qualitativa e, portanto, representado por uma ou mais variáveis *dummy* (que assumem valor 0 ou 1), dependendo da quantidade de possibilidades de resposta (categorias) desta variável dependente. Quando a variável dependente que representa o fenômeno em estudo é qualitativa, porém oferece mais de duas possibilidades de resposta (categorias), deve-se usar o método de regressão logística multinomial para estimar as probabilidades de ocorrência de cada alternativa.

De maneira geral, para um modelo em que a variável dependente assume M categorias de resposta, pode-se escrever a expressão das probabilidades p_{i_m} $(m = 0,\ 1,\ \dots\ ,\ M - 1)$ da seguinte forma:

$$p_{i_m} = \frac{e^{Z_{i_m}}}{\sum_{m=0}^{M-1} e^{Z_{i_m}}} \qquad \text{Equação (1)}$$

Sendo Z_{i_m} definido por:

$$Z_{i_m} = \alpha_m + \beta_{1m}.X_{1m} + \beta_{2m}.X_{2m} + \dots + \beta_{km}.X_{km} \qquad \text{Equação (2)}$$

Z_{i_m} é chamado logito, α representa a constante, β_j $(j = 1, 2,\dots, k)$ são os parâmetros estimados de cada variável explicativa, X_j são as variáveis explicativas e o sub índice *i* representa cada observação da amostra $(i = 1, 2,\dots, n$, em que *n* é o tamanho da amostra).

As variáveis α *e* β são determinadas buscando-se a maximização da função de verossimilhança, descrita pela equação (3), e sua significância estatística é verificada pelo teste de z de Wald. Através deste teste, confirma-se quais dos parâmetros são realmente significantes, ou seja, se influenciam significativamente na estimativa de probabilidade de um evento ocorrer, ou não.

$$LL = \sum_{i=1}^{n} \sum_{m=0}^{M-1} \left[Y_{im} . ln \left(\frac{e^{Z_{i_m}}}{\sum_{m=0}^{M-1} e^{Z_{i_m}}} \right) \right] \qquad \text{Equação (3)}$$

Assim, pode-se calcular a probabilidade de cada evento, pela equação (1). A classificação é definida para o evento de maior probabilidade calculado (Fávero, 2017).

Resumidamente, o modelo segue os seguintes passos:

1. Definir a categoria de referência.
2. Calcular as probabilidades de ocorrência de cada uma das categorias
3. Cálculo dos parâmetros das funções de regressão
4. Escrever probabilidades de ocorrência de cada uma das categorias da variável dependente
5. Classificação de cada observação, com base na maior probabilidade entre as calculadas.

O algoritmo aprendizado de máquina via regressão logística utiliza esse cálculo e otimiza o processo a cada iteração, através da regularização, uma técnica para evitar que o modelo se ajuste demais aos dados de treinamento (*overfitting*) e melhore sua capacidade de generalização (Bühlmann; Geer, 2011).

A regularização tem 2 parâmetros principais:

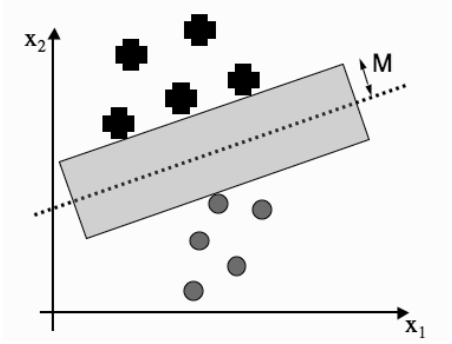
- **Tipo** (ou regra) de regularização: Tipo L1 ou L2
- **Custo**: peso dado à regra. Custos muito altos podem gerar *overfitting* e muito baixos podem gerar imprecisão no modelo.

2.5.3 Máquina de Vetores de Suporte (*Support Vector Machine*, SVM)

Marsland (2009) apresenta que o SVM é um dos algoritmos mais populares no aprendizado de máquina moderno. Frequentemente os SVM fornecem um desempenho de classificação significativamente melhor do que outros algoritmos de aprendizado de máquina para conjuntos de dados de tamanho razoável (eles não funcionam bem em conjuntos de dados extremamente grandes, pois envolvem uma inversão de matriz de dados, que é computacionalmente muito cara).

A aprendizagem de máquina de suporte busca encontrar através de cálculos, o hiperplano que melhor separa dois conjuntos de dados pertencentes a classes diferentes (Figura 6). Definindo o hiperplano com a maior margem de separação possível entre as classes, o algoritmo identifica os pontos de classificação mais próximos do hiperplano definido. Esses pontos mais próximos são chamados de vetores de suporte.

Figura 6: Margem máxima encontrada por um SVM para classificar dados.



Fonte: Marsland (2009).

Após algumas transformações e manipulações, o algoritmo aumenta a dimensão dos dados para que, mesmo que na dimensão original não fosse possível traçar uma linha, ou hiperplano, de classificação, torna-se possível ao manipular o problema num espaço de maiores dimensões. Esse processo é também conhecido como "truque do kernel". Dessa forma, o algoritmo é capaz de classificar conjuntos de dados de forma eficiente.

O SVM aplicado a classificações tem quatro parâmetros principais:

- **Custo**: quanto o modelo tem liberdade para errar.
- **Núcleo**: Função matemática utilizada para realizar as transformações dimensionais.
- **Tolerância**: Quanto o algoritmo pode se afastar da resposta correta.
- **Limite iterações**: máximo número de iterações.

2.5.4 k-Vizinhos Mais Próximo (*k-Nearest Neighbor*, k-NN)

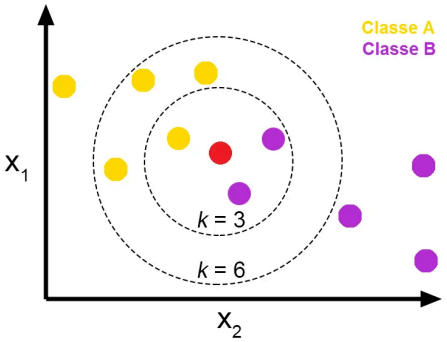
Para Russel e Norvig (2004), o algoritmo de aprendizagem de k-vizinhos mais próximos "é bastante simples de implementar, exige pouco no modo de ajuste e frequentemente funciona muito bem". O método considera que as propriedades de qualquer

ponto de entrada x específico tem propriedades semelhantes às dos seus *k* vizinhos mais próximos. Halder et al. (2024) apresentam os passos do algoritmo kNN:

1. Seleção de k: O primeiro passo é definir k, um hiperparâmetro crítico que deve ser ajustado com base nas características específicas do conjunto de dados.
2. Calcular Distâncias: Calcular a distância entre a nova instância e todos os pontos no conjunto de dados de treinamento. Métricas comuns para esse cálculo incluem distâncias Euclidiana, Manhattan e Minkowski.
3. Identificar Vizinhos Mais Próximos: Ordene todos os pontos no conjunto de treinamento do mais próximo para o mais distante do novo ponto e selecione os k pontos mais próximos.
4. Agregar Respostas Vizinhas: Para tarefas de classificação, a previsão pode ser o rótulo majoritário entre esses *k* vizinhos mais próximos (uniforme), ou receber um peso baseado na distância do ponto de previsão. Para regressão, pode ser a média ou mediana dos valores dos vizinhos.

A figura 7 ilustra a classificação realizada pelo método kNN. Na figura, o ponto central (vermelho) é escolhido ser de classe A (amarelo) ou B (roxo) a depender do valor de k, da forma como a distância é calculada, e como se agregam os vizinhos.

Figura 7: Ilustração do método de classificação kNN.



Fonte: José (2018)

Dessa forma, os principais parâmetros do método kNN são:

- **k**: quantidade de vizinhos mais próximos a se comparar.
- **Métrica**: Forma de ordenação dos k-vizinhos mais próximos.
- **Peso**: Importância de voto entre os k-vizinhos mais próximos (uniforme ou proximidade)

2.6 Métodos de avaliação para modelos de aprendizado de máquina

Salih et al (2021) destaca que após aplicar um algoritmo de classificação de treinamento, implementar um modelo e obter a saída da classificação, a última etapa é uma validação para descobrir a eficácia do modelo com base em várias métricas diferentes na fase do conjunto de dados de teste.

Rezende (2005) explica que a matriz de confusão de uma hipótese oferece uma medida efetiva do modelo de classificação, ao mostrar o número de classificações corretas *versus* as classificações preditas para cada classe. Considerando um conjunto de exemplos *T*, com *k* classificações possíveis. A matriz de confusão classifica os resultados preditos pelo algoritmo de classificação em duas dimensões: classes verdadeiras e classes preditas, para as *k* classes diferentes $\{C_1, C_2, ..., C_k)\}$. Assim pode-se calcular a quantidade de acertos e erros do modelo de aprendizagem. O número de acertos, para cada classe, se localiza na diagonal principal $M(C_i,C_i)$ da matriz. Os demais elementos $M(C_i,C_j)$, para $i \neq j$, representam erros na classificação. A matriz de confusão de um classificador ideal possui todos esses elementos iguais a zero uma vez que ele não comete erros. Na tabela 1 é ilustrada uma matriz de confusão de um modelo classificador genérico com *k* classes possíveis. A matriz de confusão de um classificador pode ser apresentada em valores absolutos ou em taxas de previsão.

Tabela 1: Matriz de confusão de um classificador genérico com k classes.

Classe	predita C_1	predita C_2	...	predita C_k
verdadeira C1	$M(C_1,C_1)$	$M(C_1,C_2)$	···	$M(C_1,C_k)$
verdadeira C2	$M(C_2,C_1)$	$M(C_2,C_2)$	···	$M(C_2,C_k)$
...	...	...	...	...
verdadeira C_k	$M(C_k,C_1)$	$M(C_k,C_2)$	···	$M(C_k,C_k)$

Fonte: Adaptada de Rezende (2005)

Para simplificação, considere-se um exemplo com apenas duas classes de classificação: Classe 1, positiva (+), e Classe 2 negativa (-). Analisando as possibilidades de respostas do algoritmo, obtém-se 4 tipos:

- **Verdadeiro Positivo** (*True Positive*, **TP**): quando o modelo indicou a classe corretamente. No exemplo, indica classe 1 (+) quando a classificação correta é classe 1(+)
- **Falso Positivo** (*False Positive*, **FP**): quando o modelo indicou uma classe, mas a classe real não era essa. No exemplo, indicou ser classe 1 (+), mas a classificação correta era classe 2 (-).
- **Falso Negativo** (*False Negative*, **FN**): quando o modelo não indica uma classe, mas essa classe era a correta. No exemplo, indica não ser classe 1 (+) quando a classificação correta é classe 1(+).
- **Verdadeiro Negativo** (*True Negative*, **TN**): quando o modelo indica corretamente que a classificação não é dada opção. No exemplo, indica não ser classe 1 (+) quando a classificação correta é classe 2 (+).

Na tabela 2 é ilustrada a forma geral da matriz de confusão para o exemplo de duas classes.

Tabela 2: Matriz de confusão para o classificação com duas classes

Classe	predita $C_{1\ (+)}$	predita $C_{2\ (-)}$
verdadeira $C_{1\ (+)}$	Verdadeiro Positivo (TP)	Falso Negativo (FN)
verdadeira $C_{2\ (-)}$	Falso Positivo (FP)	Verdadeiro Negativo (TN)

Fonte: Adaptada de Rezende (2005).

A partir dessas classificações, Stąpor (2017) define acurácia (Acc) e erro (Err), e Salih et al (2021), acrescentam outras métricas de avaliação que foram organizadas na tabela 3. Todas as métricas indicadas variam entre 0 e 1, sendo 1 o valor de máximo acerto e, portanto, desejado.

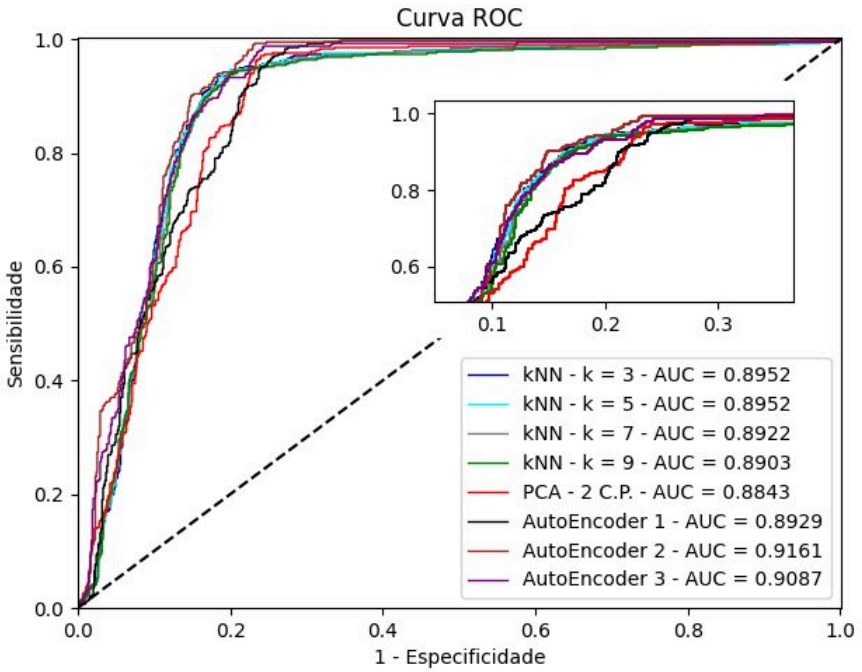
Tabela 3: Principais métricas de avaliação de desempenho de algorítmos de classificação.

Métrica	Fórmula	Descrição
Acurácia (Acc)	(TP+TN)/(TP+TN+FP+FN)	Total de classificações corretas dividido pelo total de dados classificados
Taxa de Erros (Err)	(FP + FN)/(TP + FN + TN + FP) = 1-Acc	Taxa de erros de classificação
Precisão	TP / (TP+FP)	Verdadeiros positivos que são previstos corretamente a partir do total de padrões previstos em uma classe positiva.
Taxa de Lembrança (*Recall*)	TP / (TP+FN)	Padrões positivos que são corretamente classificados como ataque
F1	2*recall*precision / (recall+precision)	Média harmônica balanceada entre *recall* e precisão
Área sob a curva (AUC)	$\int_0^1 (\frac{TP}{TP+FN} + \frac{FP}{TN+FP})$	A Área Sob a Curva mede o desempenho de uma classificação binária. É a área sob a curva ROC (*receiver-operating curve*)

Fonte: Adaptado de Salih et al (2021)

A área sob a curva característica de operação do receptor (ROC, do inglês, *receiver-operating curve*) é o gráfico da taxa de verdadeiros positivos (também chamada sensibilidade) no eixo y, versus a taxa de falsos positivos (também chamada especificidade). Cada execução do classificador gera um ponto no gráfico ROC. A curva é gerada ao se conectar os pontos (0,0) e (1,1), passando por todas as classificações realizadas. Dessa forma, o classificador perfeito, passa pelo ponto (0,1) (100% TP, 0% FP), o "anti-classificador", que erra todas as estimativas, passa pelo (1,0) e o classificador aleatório traça uma reta entre os ponto (0,0) e (1,1) - considerando que as classes positivas e negativas são igualmente comuns (Marsland, 2009). A curva ROC gera uma comparação visual entre os modelos e o cálculo da área sob sua curva, uma poderosa comparação numérica. Um exemplo de curva ROC é exemplificada na Figura 8.

Figura 8: Exemplo de curva ROC (*receiver-operating curve*).



Fonte: Chagas; Lovisolo; Tcheou (2021).

James et al. (2023) explica que não há método que funcione sempre para todos os casos, mas para cada tipo de conjunto de dados, um método vai se destacar em relação aos demais. Sheth et al (2022) indicam que os resultados podem depender muito do tamanho da base de dados e comparam diversos trabalhos que apresentam uma revisão bibliográfica na qual a maioria dos modelos de destaque atinge acima de 95% de acurácia. Entretanto, modelos que atingem 70% Acc ou 80% F1 são indicados como os melhores, de forma que não há indicação de valores típicos, mas define-se um modelo como “bom” dentro da realidade de cada conjunto de dados disponível.

3. Materiais e métodos

A metodologia adotada para a prova do conceito pode ser dividida em 3 etapas:

1. Aquisição dos dados.
2. Tratamento dos dados
3. Testes de convergência de modelos de aprendizado de máquina

3.1 Aquisição de dados

Os dados foram adquiridos por Pereira (2022) e apresentados por Porto (2023) na forma de arquivos de áudio, gravados em um smartphone Motorola Moto G5s plus, utilizando o aplicativo “Audio Recorder” para Android. A frequência de amostragem foi de 44 kHz, com uma resolução equivalente a 16 bits e em formato wav. Foram gerados 11 arquivos de áudio, 9 referentes aos testes de corte e 2 referentes ao som de fundo, quando a máquina estava ligada, mas não estava cortando.

Os 9 testes foram realizados num torno mecânico CNC Romi Centur 30D com 12,5 CV e 4.000 RPM no eixo-árvore, durante a usinagem de um cilindro de liga de titânio Ti-6Al-4V ELI com 14 mm de diâmetro e 100 mm de comprimento total. Cada um dos nove testes variou uma das seguintes parâmetros de usinagem: velocidade de rotação (n), avanço (f) e profundidade de corte (a_p). A tabela 4 mostra os parâmetros de usinagem de cada teste, bem como as condições derivadas, velocidade de corte (v_c) e velocidade de avanço (v_f).

Tabela 4: Variação das condições de usinagem para cada teste realizado.

Teste	n (rpm)	f (mm)	a_p (mm)	v_c (m/min)	v_f (mm/min)
#1	682	0,10	0,2	30	68
#2	682	0,15	0,4	30	102
#3	682	0,20	0,8	30	136
#4	1.364	0,10	0,4	60	136
#5	1.364	0,15	0,8	60	205
#6	1.364	0,20	0,2	60	273
#7	2.046	0,10	0,8	90	205
#8	2.046	0,15	0,2	90	307
#9	2.046	0,20	0,4	90	409

Fonte: Próprio autor.

3.2 Tratamento dos dados

3.2.1 Leitura, Fracionamento e FFT

O processo de leitura e fracionamento dos arquivos de áudio foi realizado no software SciLab (versão 25.0.0). Cada um dos arquivos de áudio foi inicialmente carregado no software e segmentado em intervalos de 1 segundo. Em seguida, a transformada discreta de Fourier (DFT, da sigla em inglês *Discrete Fourier Transform*) de cada segmento de 1 segundo foi calculada. Por fim, um arquivo .csv foi gerado. Dessa forma, cada segmento de cada arquivo de áudio foi processado no Scilab através do algoritmo abaixo:

I. Carregar o arquivo de áudio selecionado
II. Identificar parâmetros essenciais: taxa de amostragem e se o arquivo é mono ou stereo
III. Remoção de momentos iniciais do arquivo em silêncio
IV. Definir a duração do segmento que será utilizado
V. Segmentar o arquivo
VI. Calcular a DFT de cada segmento
VII. Gerar um arquivo .csv com a resposta da DFT para cada segmento

O programa completo pode ser conferido no Apêndice A.

Assim, foram gerados 314 arquivos de .csv com duas colunas cada. A primeira coluna sendo a frequência, de 0 a 22.049 Hz (igual para todos os arquivos) e a segunda, a magnitude do sinal para cada uma das frequências. Todos os arquivos foram reunidos numa pasta do Google Drive.

3.2.2 Compilação dos arquivos da DFT

Utilizando o Google App Script, um código foi escrito para reunir em um único arquivo a primeira coluna com as frequências e mais 314 colunas com a magnitude do sinal, para cada frequência, em cada segmento. Este código pode ser conferido no Apêndice B.

3.2.3 Formatação final do arquivo

Utilizando o Google Planilhas, os parâmetros de corte de cada teste foram adicionados e os títulos de linhas e colunas foram editados para melhorar a compreensão e um classificador foi criado para indicar se o segmento é de um momento de corte ou de som de fundo (background).

Para a formatação ideal da próxima etapa, também é necessário transpor a planilha, entretanto, neste momento, o arquivo contém 315 colunas e 22.060 linhas, mas o limite de colunas do Google Planilhas é de 18.278. Para contornar essa limitação, o arquivo foi dividido em duas partes, unido com o software Orange Data Mining (versão 3.38.1) e salvo como .csv, que não tem nenhum tipo de limitação de linhas ou colunas.

As primeiras colunas e linhas do arquivo são mostradas na tabela 5 para exemplificar sua formatação final.

Tabela 5: Conceito da formatação do arquivo final de dados.

Teste	Fragmento	Tipo	vc (m/min)	f (mm)	ap (mm)	n (rpm)	vf (mm/min)	tc (s)	Mag. Freq=0	Mag. Freq=1	Mag. Freq=...
Test001	ID 001	Cortando	30	0.1	0.2	682	68	88	0.0777587891	0.6789827193	…
Test001	ID 002	Cortando	30	0.1	0.2	682	68	88	0.4954223633	0.3116661431	…
Test001	ID 003	Cortando	30	0.1	0.2	682	68	88	0.8996276855	0.7637475279	…
Test001	ID 004	Cortando	30	0.1	0.2	682	68	88	0.6172485352	0.3013205932	…
Test001	ID 005	Cortando	30	0.1	0.2	682	68	88	0.0321350098	0.1998608838	…
Test001	ID 006	Cortando	30	0.1	0.2	682	68	88	0.0852661133	0.3696560746	…
Test001	ID 007	Cortando	30	0.1	0.2	682	68	88	0.732421875	1.004703989	…
Test001	ID 008	Cortando	30	0.1	0.2	682	68	88	0.2944335938	0.9678923864	…
Test001	ID 009	Cortando	30	0.1	0.2	682	68	88	1.628936768	1.993394063	…
Test001	ID 010	Cortando	30	0.1	0.2	682	68	88	1.029541016	0.7480343093	…
Test001	ID 011	Cortando	30	0.1	0.2	682	68	88	0.2522583008	0.9427847084	…
…	…	…	…	…	…	…	…	…	…	…	…

Fonte: Próprio autor

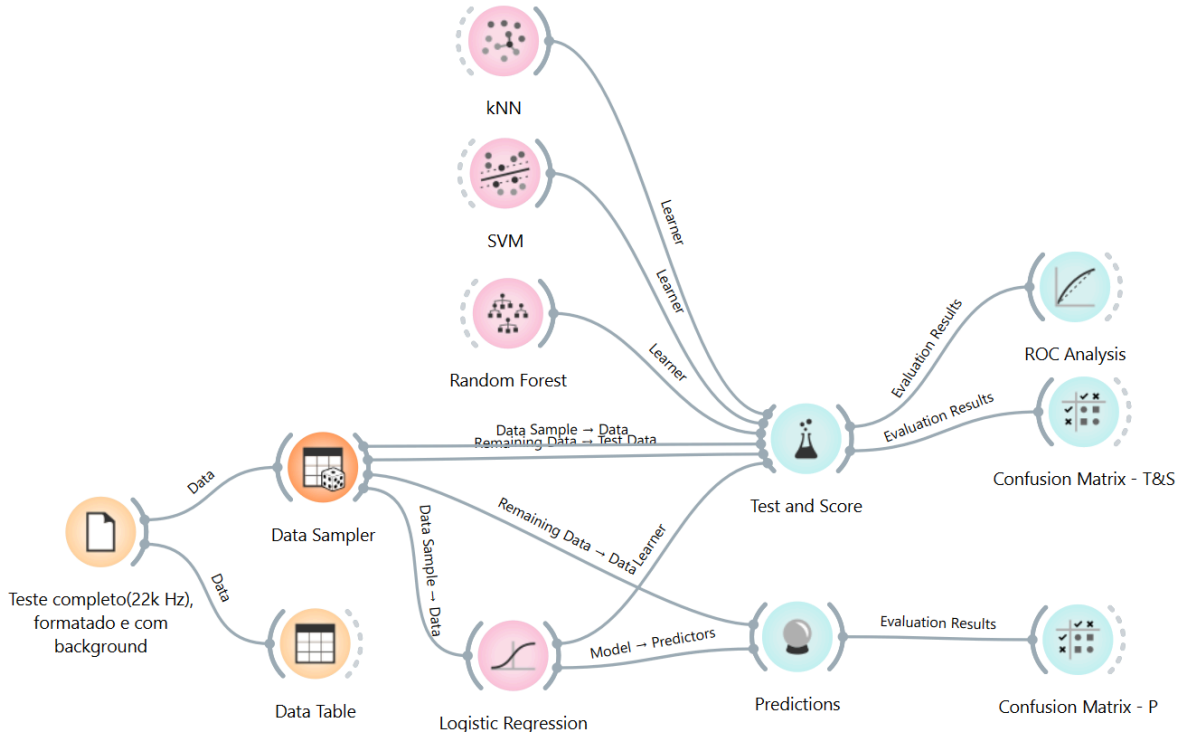
3.3 Testes de convergência de modelos de aprendizado de máquina

Para o aprendizado de máquina, foi utilizado o software Orange Data Mining na versão 3.38.1. O esquema de conexão dos *widgets* é apresentado na Figura 9. Nela, pode ser observado o fluxo de trabalho realizado no Orange.

O fluxo começa com a importação dos dados através do *widget* nomeado “Teste completo (22k Hz), formatado e com background”. Os dados importados são conectados ao “*Data Table*” para verificar a integridade e apresentação dos dados inseridos. Os dados importados também são conectados ao “*Data Sampler*”, cuja função é dividir os dados aleatoriamente numa proporção definida de forma a separarmos dados para treinamento dos algoritmos e dados de testes da sua performance. Os dados de teste e os dados de treinamento são conectados ao *widget* “*Test and Learn*”, que se conecta aos modelos de aprendizado de máquina escolhidos. Conectou-se à saída de “*Test and Learn*” os *widgets* “*ROC Analysis*” e “*Confusion Matrix - T&L*” para complementar a avaliação e comparação dos modelos. Por

fim, o modelo de melhor performance, regressão logística, recebeu os dados de treino para poder se conectar ao widget "*Predictions*", que já recebia os dados de teste. "Predictions" foi conectado à "Confusion Matriz - P" para validação final do modelo.

Figura 9: Esquema de conexão dos widgets do Orange Data Mining.



Fonte: Próprio autor

3.3.1 Definição de colunas

As colunas foram classificadas nos papeis e tipos, conforme descrito na Tabela 6. A coluna principal é a Teste, definida como target, do tipo categoria, pois é ela que o modelo de aprendizagem buscará identificar corretamente. Ela apresenta a classificação entre teste 1, teste 2, … , teste 9, background 1 e background 2.

Tabela 6: Classificação das colunas de dados usada no Orange Data Mining.

Nome da Coluna	Teste	Fragmento	Tipo	vc (m/min)	f (mm)	ap (mm)	n (rpm)	vf (mm/min)	tc (s)	Mag Freq= ...
Papel	target	meta	feature	meta	meta	meta	meta	meta	meta	feature
Tipo	categorical	text	categorical	numeric	numeric	numeric	numeric	numeric	numeric	numeric

Fonte: Próprio autor

As colunas Fragmento, vc, f, ap, n, vf e tc, representam uma identificação do fragmento, e as condições de usinagem. Essas colunas foram classificadas como meta dados, não sendo utilizadas diretamente na aprendizagem dos algoritmos. Entretanto elas ficam listadas como informações úteis para comparação e análise posterior.

Por fim, a coluna Tipo indica se o arquivo se referia à situação em corte ou *background*, e as 22.050 colunas "Mag Freq" indicam a magnitude do sinal sonoro para cada uma das frequências. Tanto a coluna Tipo, quanto as "Mag Freq", foram classificadas como *feature*, significando que são essas informações que os algoritmos vão utilizar para aprender quais características devem separar um teste do outro.

3.3.2 Data Sampler

No software Orange, o widget *Data Sampler* foi utilizado para selecionar uma parte aleatória dos dados para o treino dos modelos. Foi fixada uma proporção de 75% dos dados para o treinamento dos modelos e 25% para a validação final do modelo.

As saídas *Data Sampler* e *remaining rata* do *Data Sampler* foram conectadas às entradas de *Data* e *Test Data* do *widget Test*, respectivamente.

3.3.3 Modelos de aprendizado de máquina

Os modelos SVM, kNN, Floresta Aleatória e Regressão Logística foram selecionados por trazerem tipicamente boas respostas em classificação de dados. Após a realização de diversos testes para se buscar maximizar os resultados obtidos com cada um dos modelos, os parâmetros principais para cada um deles foi definido, como se apresenta abaixo:

- SVM
    - Custo: 1
    - Núcleo: RBF (*radial basis function*)
    - Tolerância numérica: 10^{-4}
    - Limite de interações 100
- k-vizinho mais próximo (kNN)
    - k = 5
    - Métrica: Euclidiana (distância entre os pontos)
    - Peso: Uniforme
- Floresta Aleatória
    - Número de Árvores: 10

    - Não dividir em subgrupos menores do que: 5
- Regressão Logística
    - Regularização: Lasso (L1)
    - Força: C=35

3.3.4 Validação dos modelos de aprendizados de máquina

Utilizando o *widget Test and Score*, uma matriz de confusão foi gerada para cada modelo e foram avaliados os melhores valores de Acurácia, Área Sob a Curva ROC (AUC), F1-score, Precisão e Recall.

Por fim, o melhor modelo foi validado com o uso do *widget Predictions* e uma matriz de confusão.

4. Resultados e discussão

A tabela 7 mostra a duração de cada arquivo de áudio, bem como a quantidade de segmentos criados para cada arquivo, de cada teste e dos arquivos de background. Adicionou-se o tempo de corte (t_c), a velocidade de rotação (n) e o avanço (f) para observar-se que os primeiros testes apresentam maiores durações devido a ocorrerem em maior tempo de corte. O tempo de corte, por sua vez, é influenciado pela velocidade de rotação (inversamente proporcional) e pelo avanço (diretamente proporcional).

Tabela 7: Duração dos arquivos de áudio e quantidade de segmentos criados.

Teste	Duração (s)	Segmentos Criados	t_c (s)	n (rpm)	f (mm)
#1	72	72	88	682	0,10
#2	48	48	59	682	0,15
#3	33	33	44	682	0,20
#4	35	35	44	1.364	0,10
#5	25	25	29	1.364	0,15
#6	21	21	22	1.364	0,20
#7	28	28	29	2.046	0,10
#8	21	21	20	2.046	0,15
#9	19	17	15	2.046	0,20
Background 1	5	5	-	-	-
Background 2	7	7	-	-	-

Fonte: Próprio Autor

Apenas para o teste 9 foram identificados alguns segundos do arquivo de áudio que estavam em completo silêncio, e sem informação. Por isso, o algoritmo de criação dos segmentos (ver 3.3.1) identificou e removeu 2 segundos deste arquivo, gerando assim 2 segmentos a menos do que a quantidade de segundos da duração do arquivo.

4.1 Comparação dos modelos

A tabela 8 apresenta os resultados nas métricas de desempenho avaliadas. Todos os modelos obtiveram resultados considerados bons, acima de 80% Acc e Precisão, além de AUC > 90%. Entretanto, o modelo de regressão logística, mesmo sendo um modelo bastante simples, foi o de maior eficácia na classificação dos testes realizados e, por conseguinte, das condições de usinagem. Ele apresentou 96,2% de acurácia, 96,4% de precisão e 99,75% em AUC, superando os demais em todas as métricas avaliadas.

Tabela 8: Comparação dos resultados dos modelos de aprendizado de máquina.

Modelo	Acurácia (ACC)	F1	Precisão	Taxa de Lembrança (Recall)	Área sob a curva (AUC)
SVM	80,08%	79,31%	80,94%	80,08%	93,49%
kNN	90,68%	90,99%	92,27%	90,68%	98,29%
Floresta Aleatória (Random Forest)	84,32%	84,42%	85,80%	84,32%	95,16%
Regressão Logística (Logistic Regression)	96,19%	96,22%	96,42%	96,19%	99,75%

Fonte: Próprio autor

As figuras 10 a 13 apresentam as matrizes de confusão, com as proporções de predições, geradas após passagem pelo *widget Test&Learn* para os modelos testados. Nelas, pode-se observar o acerto de cada modelo, para cada conjunto de dados.

Figura 10: Matriz de confusão com as proporções de predições para SVM

Predicted

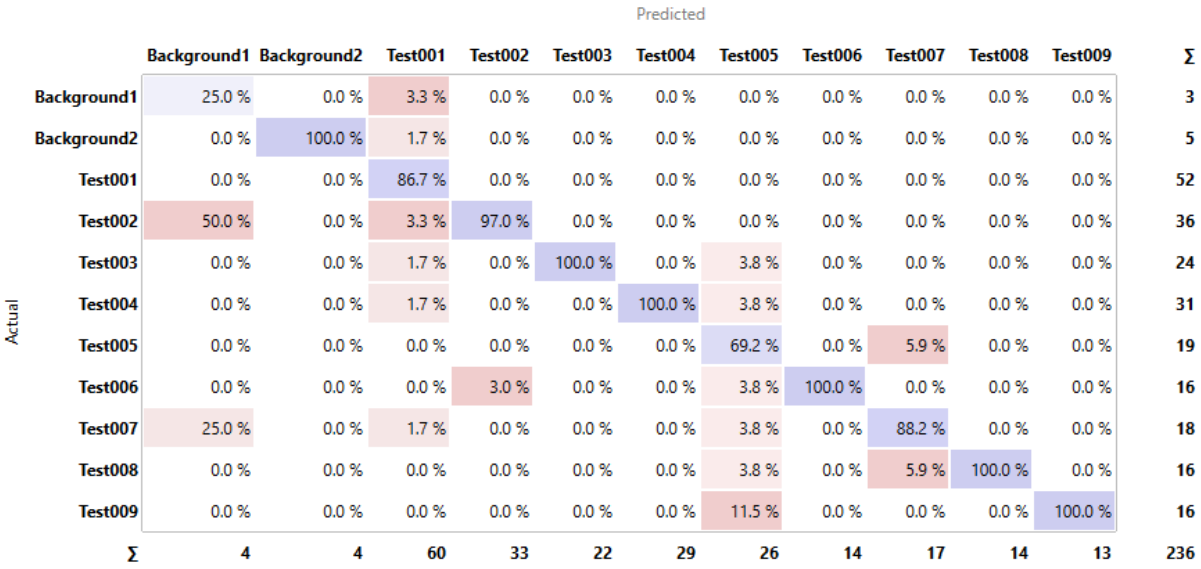
Actual	Background1	Background2	Test001	Test002	Test003	Test004	Test005	Test006	Test007	Test008	Test009	Σ
Background1	NA	NA	4.2 %	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	3
Background2	NA	NA	6.9 %	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	5
Test001	NA	NA	69.4 %	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	8.0 %	52
Test002	NA	NA	4.2 %	93.9 %	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	8.0 %	36
Test003	NA	NA	1.4 %	3.0 %	95.0 %	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	12.0 %	24
Test004	NA	NA	0.0 %	0.0 %	0.0 %	96.7 %	0.0 %	0.0 %	0.0 %	0.0 %	8.0 %	31
Test005	NA	NA	5.6 %	0.0 %	0.0 %	3.3 %	100.0 %	0.0 %	0.0 %	0.0 %	4.0 %	19
Test006	NA	NA	2.8 %	0.0 %	5.0 %	0.0 %	0.0 %	76.5 %	0.0 %	0.0 %	0.0 %	16
Test007	NA	NA	2.8 %	0.0 %	0.0 %	0.0 %	0.0 %	5.9 %	92.9 %	0.0 %	8.0 %	18
Test008	NA	NA	1.4 %	0.0 %	0.0 %	0.0 %	0.0 %	11.8 %	7.1 %	83.3 %	8.0 %	16
Test009	NA	NA	1.4 %	3.0 %	0.0 %	0.0 %	0.0 %	5.9 %	0.0 %	16.7 %	44.0 %	16
Σ	0	0	72	33	20	30	13	17	14	12	25	236

Fonte: Próprio autor

Na figura 10, é possível observar que o modelo SVM teve bastante dificuldade no teste 9, provavelmente por este teste ter uma quantidade pequena de dados, mas também teve certa dificuldade nos testes 6, 8 e 1. Após o teste 9, os testes 6 e 8 são os menores, com um total de 21 segmentos, indicando novamente dificuldade do algoritmo com a quantidade de dados Entretanto, o teste 1, que tinha a maior

quantidade de dados de treinamento disponíveis, não teve o maior índice de acertos. Pelo volume maior de dados, o algoritmo indicou o teste 1 para outros testes erroneamente diversas vezes, demonstrando dificuldade em filtrar as características determinantes para a classificação correta deste teste.

Figura 11: Matriz de confusão com as proporções de predições para kNN

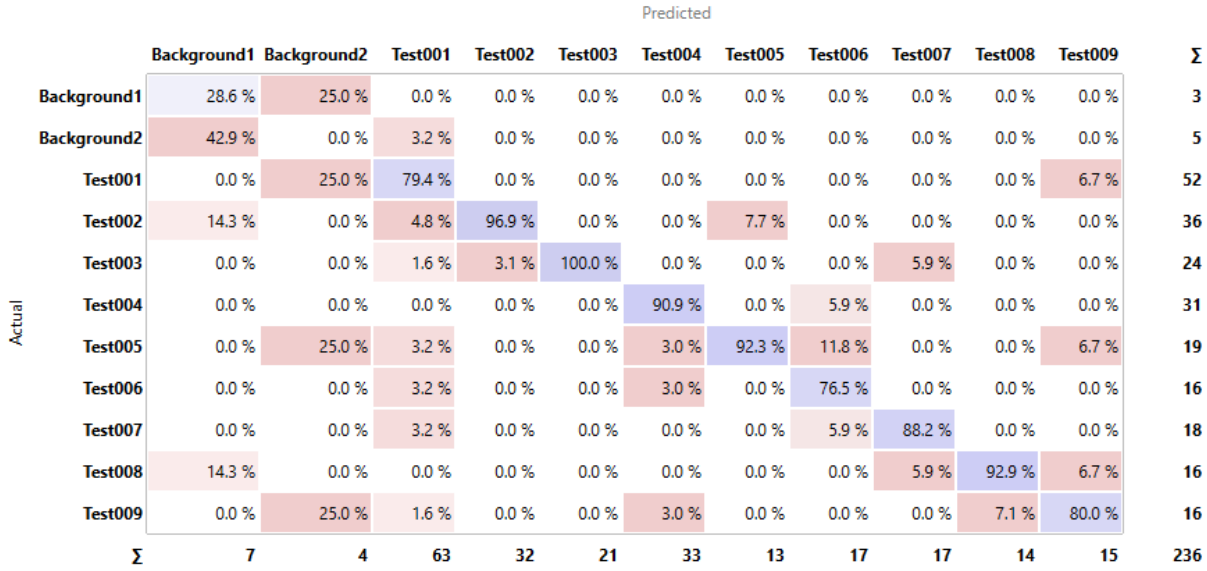


Fonte: Próprio autor

O modelo kNN, o segundo a melhor classificar os dados, apresentou Acc 90,68%, tendo facilidade em identificar a maioria dos testes. Ele teve mais dificuldade em identificar os testes 5, 1 e 7, em ordem decrescente de acerto. Nos três casos, o algoritmo distribuiu e confundiu as respostas entre esses testes. No teste 9, mesmo o com menor quantidade de amostras, o modelo teve 100% de acerto dos casos, entretanto teve dúvidas se o teste 5 era na verdade o 9, sendo que estes dois testes não compartilhavam nenhum parâmetro em comum.

Na figura 12, que retrata a matriz de confusão do modelo de floresta aleatória, observa-se maior volume de erros, concentrados nos testes 6, 1 e 9, em ordem decrescente de acerto, mas com considerável dispersão dos erros por toda a matriz. Novamente, o primeiro e nono testes, com maior e menor volumes de dados, respectivamente, se mostraram um desafio para o modelo.

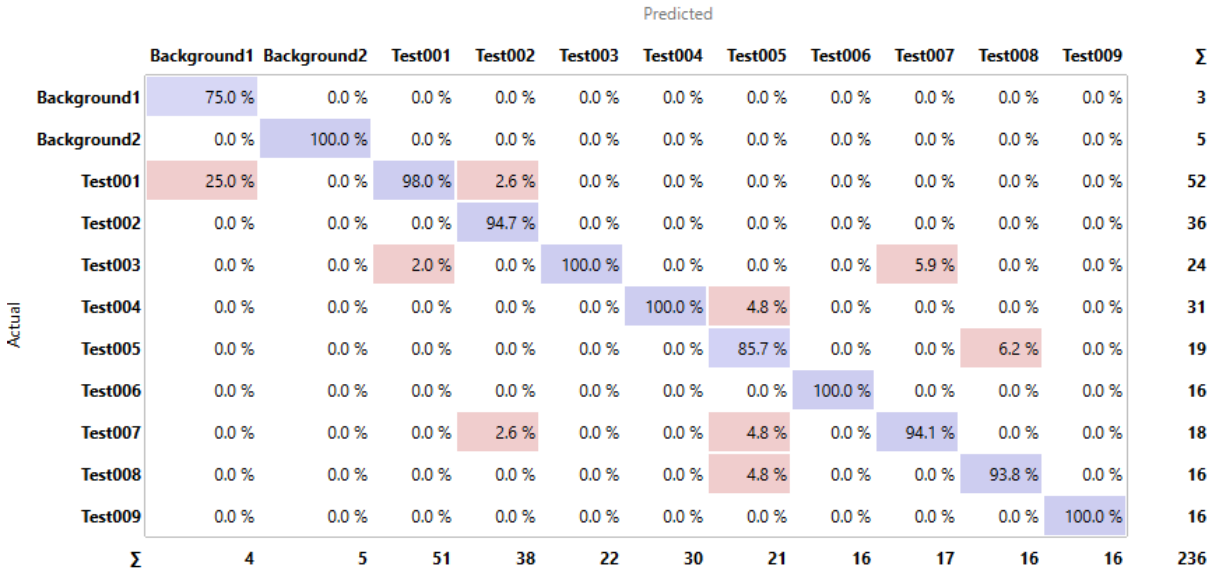
Figura 12: Matriz de confusão com as proporções de predições para Floresta Aleatória



Fonte: Próprio autor

A matriz de confusão do modelo de regressão logística é apresentada na figura 13. Destaca-se o alto nível de precisão para todos os testes, tendo como acerto mínimo o teste 5, com 85,7% de acerto. O modelo não se mostrou impactado pela discrepância no volume de dados, gerando previsões bastante acertadas.

Figura 13: Matriz de confusão com as proporções de predições para Regressão Logística

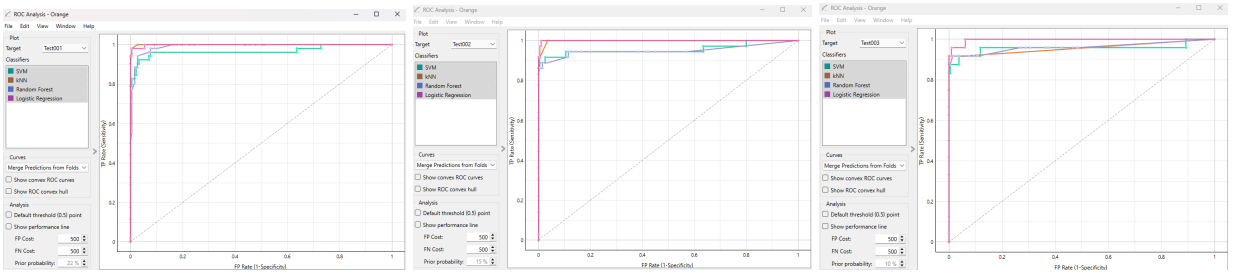


Fonte: Próprio autor

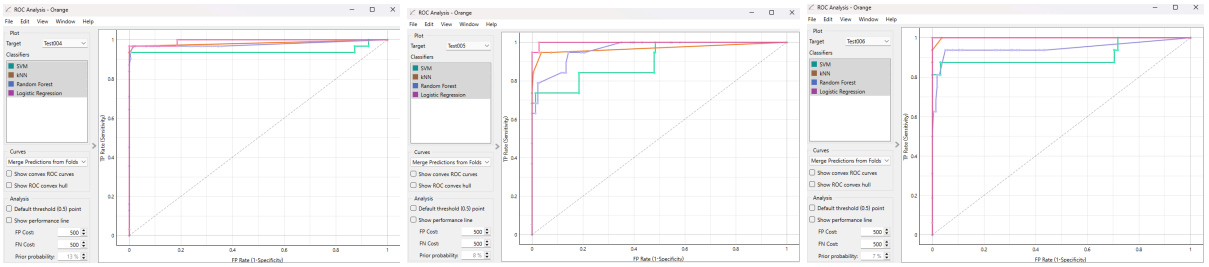
A figura 14 apresenta as curvas ROC para cada um dos 9 testes. A ROC evidencia que, apesar da dificuldade em acertar o primeiro teste, a taxa de acerto nele, que tem mais dados que os demais foi, no geral dos modelos avaliados, maior do que para os testes 5 a 9, quando a quantidade de segmentos disponíveis fica abaixo de 30.

É possível perceber como o modelo SVM (linha verde) teve mais dificuldades, especialmente no teste 9, em contraste com o modelo de regressão logística (linha rosa), que apresenta uma curva quase perfeita na maioria dos testes.

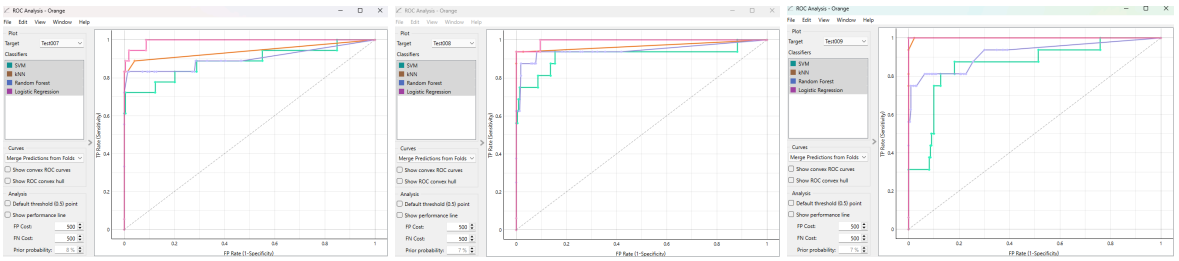
Figura 14: Curva ROC para os 9 testes realizados.



(a) Teste 1 (b) Teste 2 (c) Teste 3



(d) Teste 4 (e) Teste 5 (f) Teste 6



(g) Teste 7 (h) Teste 8 (i) Teste 9

Fonte: Próprio autor.

Os resultados demonstram que a regressão logística, mesmo sendo o modelo mais simples, foi o de maior eficácia na classificação dos testes realizados e, por conseguinte, das

condições de usinagem. Com 96,2% de acurácia, 96,4% de precisão e 99,75% em AUC, o modelo superou os demais em todas as métricas avaliadas.

A análise das matrizes de confusão e dos gráficos ROC explicita como o modelo de regressão logística consegue prever com alta precisão, mesmo os testes 5 a 9, nos quais os demais modelos tiveram maior dificuldade.

Apesar dos dados bastante satisfatórios obtidos com as altas taxas de Acc e AUC, os dados analisados evidenciam duas oportunidades de melhorias para todo o processo:

1. Aumentar a quantidade de segmentos (aumentando a duração dos cortes de teste ou reduzindo o tamanho do segmento do arquivo de áudio)
2. Homogeneizar a quantidade de segmentos disponíveis para cada teste, reduzindo as chances de influenciar a decisão pelo simples volume maior de amostras.

Essas duas otimizações no processo de pré-processamento devem melhorar o desempenho geral dos algoritmos de machine learning e reduzir o impacto negaria na performance que condições externas ao experimento possam ter na correta previsão de condições de usinagem.

4.2 Confirmação do modelo de regressão logística

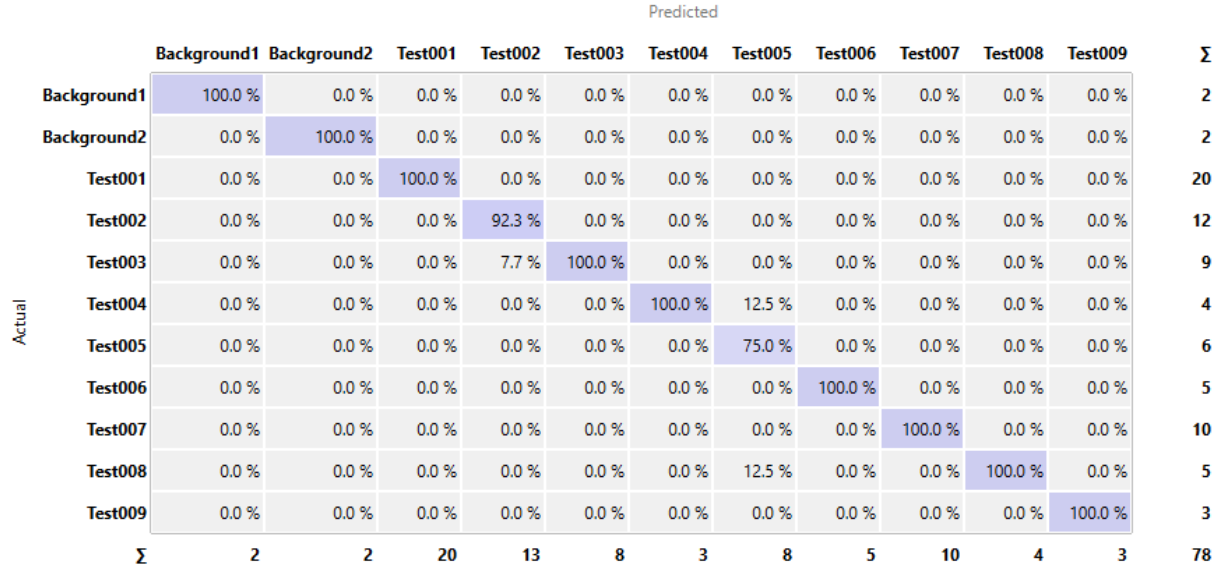
A verificação da capacidade preditiva do modelo de regressão logística foi verificada com o widget *Predictions*, produzindo resultados iguais ou ligeiramente melhores dos previstos pelos testes anteriores. A tabela 9 compara os resultados obtidos com o *Test&Lear* e *Predictions*. A pequena diferença acontece pois o *Test&Learn* trabalha com diferentes simulações (3 na configuração utilizada) com subgrupos dos dados, enquanto que o *Predictions* apenas calcula os resultados com os dados inseridos, no caso os dados não utilizados no treinamento do modelo. A figura 15 apresenta a matriz de confusão com as proporções de predições do modelo de regressão logística, na validação do *widget Predictions*.

Tabela 9: Comparação dos resultados de teste e validação do modelo de regressão logística.

Regressão Logística	Acurácia (ACC)	F1	Precisão	Taxa de Lembrança (Recall)	Área sob a curva (AUC)
Test&Learn	96,19%	96,22%	96,42%	96,19%	99,75%
Predictions	96,20%	96,20%	96,60%	96,20%	99,90%

Fonte: Próprio Autor

Figura 15: Matriz de confusão com a proporções de predições do modelo de regressão logística, na validação do widget Predictions



Fonte: Próprio Autor

A comparação entre *Test&Lear* e *Predictions* mostra grande proximidade entre eles, mostrando a capacidade de generalização de predições do modelo de regressão logística, construído neste trabalho. Entretanto, cabe destacar que o modelo foi capaz de excelentes predições num conjunto de dados que pode ser considerado próximo (segmentos diferentes de um mesmo arquivo de áudio). Um próximo passo de confirmação do poder de classificação deste modelo passaria por realizar novas gravações, em condições semelhantes, para realmente provar o poder de classificação do algoritmo.

5. Considerações finais

Utilizando o software Orange Data Mining, com apoio do SciLab e Google Workspace, foi possível testar 4 modelos de aprendizagem de máquina diferentes com o objetivo de classificar as condições de usinagem do processo de torneamento, sendo os dados capturados via áudio de um aparelho celular. Um modelo de aprendizagem de máquina capaz de classificar satisfatoriamente os testes realizados e suas condições foi obtido com sucesso. As análises indicaram o modelo de regressão logística como o mais capaz de classificar corretamente o sinal pré-processado, com AUC de 99,75%.

O maior desafio encontrado neste trabalho foi o pré-processamento dos dados, antes mesmo que eles pudessem ser avaliados pelos modelos de inteligência artificial. Este processo necessitou de 2 ferramentas diferentes e alguns dias de trabalho para atingir o formato ideal.

Esta prova de conceito, realizada com apenas 9 testes, em um único ensaio, abre portas para o desenvolvimento de ferramentas poderosas, capazes de monitorar através do áudio processos de torneamento, e usinagem de forma geral, indicando parâmetros e qualidade de processos de forma precisa e a baixo custo.

Futuros trabalhos podem se aprofundar na programação, construindo um único programa, em python talvez, capaz de realizar todos os processos deste trabalho em uma única etapa, possivelmente a partir de um dispositivo móvel. Devem ser considerados como melhorias também, uma avaliação do volume de entradas, buscando ficar acima de 30 - que se mostrou uma quantidade significativa neste trabalho - para todos os teste, e a homogeneização do volume de segmentos por teste apresentados a cada algoritmo de aprendizado de máquina.

Uma segunda possível linha de melhorias seria a utilização de modelos mais complexos para avaliar mais de um target. Utilizando também um modelo de regressão permitindo extrapolar correlações entre aumento ou diminuição de um dos parâmetros de usinagem, o modelo utilizaria o som para definir diretamente os valores dos parâmetros de usinagem, tornando-se mais amplo ao poder extrapolar suas previsões para valores não apresentados nos testes realizados.

Uma terceira opção de desenvolvimento seria a realização de mais testes, variando-se por exemplo, a ferramenta de corte e o material usinado, para construir um de bancos de dados amplo o suficiente para permitir a criação de um modelo capaz de prever os parâmetros de usinagem de uma gama maior de processos.

Referências

BLACK, J. T.; KOHSER, Ronald A. **DeGarmo's Materials and Processes in Manufacturing**. 10th. ed. United States of America: John Wiley & Sons, Inc, 2008.

BÜHLMANN, Peter; GEER, Sara van de. **Statistics for high-dimensional data: methods, theory and applications.** Berlin Heidelberg: Springer, 2011.

CHAGAS, Amanda De Oliveira Sabino; LOVISOLO, Lisandro; TCHEOU, Michel. **Detecção de Risco em Ambiente Hospitalar a partir da Informação do Estado do Canal Sem-Fio**. *In*: CONGRESSO BRASILEIRO DE INTELIGÊNCIA COMPUTACIONAL. Anais do 15. Congresso Brasileiro de Inteligência Computacional. SBIC, 1 jan. 2021. Disponível em: <https://sbic.org.br/eventos/cbic_2021/cbic2021-78/>. Acesso em: 28 jul. 2025

DA COSTA, Matheus Moreira; VAZZOLER, Marcia Regina; RIGOLIN, Guilherme Vazzoler. **Aplicações De Ferramentas De Inteligência Artificial Na Otimização De Processos Industriais**. 8º Congresso Internacional Multidisciplinar. jul.2024. Disponível em: <artigo-acbefaf59ecb97bb449391225b735a34e6d44d860dbf1a5eb40563ce-arquivo.pdf>. Acesso em: 3 jul. 2025

FÁVERO, Luiz. **Manual de análise de dados: estatística e modelagem multivariada com excel, SPSS e stata.** [S.l.]: Elsevier, 2017.

FERRAZ JR., Fábio. **Desenvolvimento de um Sistema de Monitoramento e Supervisão para o Processo de Torneamento.** Dissertação (Mestrado em Engenharia Mecânica)—São Carlos, SP: Escola de Engenharia de São Carlos da Universidade de São Paulo, 2002.

HALDER, Rajib Kumar et al. **Enhancing K-nearest neighbor algorithm: a comprehensive review and performance analysis of modifications.** Journal of Big Data, v. 11, n. 1, 11 ago. 2024.

HEIDEMAN, Michael T.; JOHNSON, Don H.; BURRUS, C. Sidney. **Gauss and the history of the fast Fourier transform.** Archive for History of Exact Sciences, v. 34, n. 3, p. 265–277, 1985.

JAMES, Gareth *et al.* **An Introduction to Statistical Learning**. Cham: Springer International Publishing, 2023.

JOSÉ, Italo. **KNN (K-Nearest Neighbors) #1**. **Medium**, 1 jul. 2018. Disponível em: <https://medium.com/brasil-ai/knn-k-nearest-neighbors-1-e140c82e9c4e>. Acesso em: 22 jul. 2025

LUDERMIR, Teresa Bernarda. **Inteligência Artificial e Aprendizado de Máquina: estado atual e tendências**. Estudos Avançados, v. 35, n. 101, p. 85–94, abr. 2021.

MACHADO, Álisson Rocha et al. **Teoria da usinagem dos materiais.** 3. ed. São Paulo: Blucher, 2015. E-book. Disponível em: https://plataforma.bvirtual.com.br. Acesso em: 6 jul. 2025.

MARSLAND, Stephen. **Machine learning: an algorithmic perspective**. Boca Raton: CRC Press, 2009.

MUTHUKRISHNAN, N. et al. **Brief History of Artificial Intelligence**. Neuroimaging Clinics of North America, v. 30, n. 4, p. 393–399, nov. 2020.

PERDIGÃO, Iago Luiz et al. **O Papel da IA na Otimização de Processos Industriais.** Revista FT, v. 27, n. 127, 27 out. 2023.

Receiver operating characteristic. Disponível em: <https://en.wikipedia.org/wiki/Receiver_operating_characteristic>. Acesso em: 10 jul. 2025.

REZENDE, Solange Oliveira (org.). **Sistemas Inteligentes: Fundamentos e Aplicações**. [S.l.]: Editora Manole, 2005.

RUSSELL, Stuart; NORVIG, Peter. **Inteligência artificial: tradução da segunda edição**. Rio de Janeiro: Elsevier : Campus, 2004.

SALIH, Azar Abid; ABDULAZEEZ, Adnan Mohsin. **Evaluation of Classification Algorithms for Intrusion Detection System: A Review.** Journal of Soft Computing and Data Mining, v. 02, n. 01, 15 abr. 2021.

Share of economic sectors in the global gross domestic product (GDP) from 2014 to 2024. Disponível em: <https://www.statista.com/statistics/256563/share-of-economic-sectors-in-the-global-gross-domestic-product/.>. Acesso em: 20 jul. 2025.

SICHMAN, J. S. **Inteligência Artificial e sociedade: avanços e riscos**. Estudos Avançados, v. 35, n. 101, p. 37–50, abr. 2021.

STĄPOR, Katarzyna. **Evaluation of classifiers: current methods and future research directions**. In: 2017 FEDERATED CONFERENCE ON COMPUTER SCIENCE AND INFORMATION SYSTEMS. Annals of Computer Science and Information Systems. PTI, 24 set. 2017. Disponível em: <https://annals-csis.org/Volume_12/drp/530.html>. Acesso em: 9 jul. 2025

VALDATI, A. DE B. **Inteligência artificial - IA**. [s.l.] Contentus, 2020.

Apêndice A - Código SciLab para processar os arquivos de áudio

```
//-------------------------------------------------------------------------
//--------  FRACIONA CADA ARQUIVO DE ÁUDIO E CALCULA A FFT  -------------
//-------------------------------------------------------------------------

//---- Carregar o arquivo WAV -----

// 1. Define o caminho para o arquivo WAV
audio_file_path = 'C:\Users\rafeal\Documents\TCC_data_sound\test_009.wav';

// 2. Carregar o arquivo de áudio
[audio_data, audio_info] = loadwave(audio_file_path);

// Verifica se o arquivo foi carregado corretamente
if isempty(audio_data) then
   disp('Erro ao carregar o arquivo de áudio. Verifique o caminho e o formato (deve ser .wav).');
   return;
end

//---- Identificação Parâmetros -----

// 3. Extrair a taxa de amostragem (Fs)
Fs = audio_info(3);
disp(sprintf('Taxa de Amostragem (Fs): %d Hz', Fs));

// 4. Selecionar um canal para análise (se for estéreo, pegamos o primeiro)
if size(audio_data, 1) > 1 then
   signal_original = audio_data(1, :); // Pega o primeiro canal (mono)
   disp('Áudio estéreo detectado. Analisando o primeiro canal.');
else
   signal_original = audio_data; // Áudio já é mono
end

// --- Remover amostras iniciais zeradas ---
threshold = 1e-5;
first_non_zero_index = find(abs(signal_original) > threshold, 1);

if isempty(first_non_zero_index) then
   disp('O arquivo de áudio parece ser todo silêncio ou muito curto para análise.');
   return;
end

signal = signal_original(first_non_zero_index:$);
```

```
disp(sprintf('Sinal original tinha %d amostras.', length(signal_original)));
disp(sprintf('Sinal após remover silêncio inicial tem %d amostras.', length(signal)));

//---- Definições segmentação -----

// Defina a duração de cada segmento em segundos
segment_duration_seconds = 1;

// Calcule o número de amostras por segmento
samples_per_segment = Fs * segment_duration_seconds;
disp(sprintf('Cada segmento terá %d amostras (equivalente a %d segundo(s)).', samples_per_segment, segment_duration_seconds));

// Obtenha o número total de amostras no sinal limpo
total_samples = length(signal);

//---- Segmentação do arquivo -----

// Calculo do número total de segmentos completos que podemos criar
num_segments = floor(total_samples / samples_per_segment);
disp(sprintf('Serão criados %d segmento(s) completo(s) de %d segundo(s).', num_segments, segment_duration_seconds));

//---- Definições criação do arquivo .CSV -----

// Definição do diretório onde os arquivos CSV da FFT serão salvos
output_directory_csv = 'C:\Users\rafael\Documents\TCC_data_sound\Fracionados e FFT';

// Defina o delimitador para o arquivo CSV (vírgula é o padrão para CSV)
csv_delimiter = ','; // Ou ';', se sua configuração regional usar ponto e vírgula como delimitador CSV

//---- Cálculo FFT e criação do arquivo -----

// Loop para criar, calcular FFT e exportar cada segmento
for i = 0:(num_segments - 1)

    // Defina os índices de início e fim para o segmento atual
    start_index = i * samples_per_segment + 1;
    end_index = (i + 1) * samples_per_segment;

    // Extraia o segmento do sinal
    current_segment = signal(start_index:end_index);

    // --- Calcular a FFT para o segmento atual ---
    N_segment = length(current_segment);
```

```
    Y_segment = fft(current_segment);

    // Obter a magnitude do espectro (apenas a primeira metade, frequências positivas)
    magnitude_spectrum_segment = abs(Y_segment(1:N_segment/2));

    // Criar o vetor de frequência para este segmento
    f_segment = (0:(N_segment/2)-1) * (Fs/N_segment);

    // --- Exportar a Magnitude do Espectro para um arquivo CSV ---
    // A única mudança aqui é a extensão .csv no nome do arquivo
    filename = sprintf('test009_%03d.csv', i + 1);
    filepath = fullfile(output_directory_csv, filename);

    fd = mopen(filepath, 'w');

    if fd == -1 then
        disp(sprintf('Erro ao abrir o arquivo: %s', filepath));
    else
        // Escreve os dados do espectro (Frequência e Magnitude)
        // Usamos o 'csv_delimiter' definido
        mfprintf(fd, 'Frequencia (Hz)%sMagnitude\n', csv_delimiter); // Cabeçalho sem o '#'
        for j = 1:length(magnitude_spectrum_segment)
            mfprintf(fd, '%.10f%s%.10f\n', f_segment(j), csv_delimiter, magnitude_spectrum_segment(j));
        end
        mclose(fd);
        disp(sprintf('Espectro do segmento %d salvo em: %s', i + 1, filepath));
    end
end
disp('Processo de divisão, cálculo de FFT e exportação para CSV concluído!');
```

Anexo B - Código App Script compilação arquivos FFT

```
function consolidateCsvColumns() {
  // --- CONFIGURAÇÕES ---
  const FOLDER_ID = '1qz1YGBpr1WFyQ63K7lV1MT8fTryDMKNR'; // <-- SUBSTITUA
PELO ID DA SUA PASTA NO GOOGLE DRIVE
  const OUTPUT_SHEET_NAME = 'Dados_FFT'; // Nome da aba onde os dados serão
consolidados

  // Se você quer criar uma nova planilha, defina CREATE_NEW_SPREADSHEET
como true.
  // Se você quer usar uma planilha existente, defina como false e preencha
o SPREADSHEET_ID_EXISTING.
  const CREATE_NEW_SPREADSHEET = true;
  const SPREADSHEET_ID_EXISTING =
'1jheyqweVbWZrFpVs-b_6x-a-iHpdcf8UlYIk9n0zKk0'; // <-- APENAS SE
CREATE_NEW_SPREADSHEET FOR FALSE

  // --- NÃO É NECESSÁRIO MODIFICAR O CÓDIGO ABAIXO DESTA LINHA ---

 let spreadsheet;
  if (CREATE_NEW_SPREADSHEET) {
    spreadsheet = SpreadsheetApp.create('Consolidado CSV por Coluna');
    Logger.log('Nova planilha criada: ' + spreadsheet.getUrl());
  } else {
    try {
      spreadsheet = SpreadsheetApp.openById(SPREADSHEET_ID_EXISTING);
      Logger.log('Usando planilha existente: ' + spreadsheet.getUrl());
    } catch (e) {
      Logger.log('Erro ao abrir planilha existente. Verifique o ID. Erro: '
+ e.toString());
      return; // Interrompe a execução se a planilha não puder ser aberta
    }
  }

  const folder = DriveApp.getFolderById(FOLDER_ID);
  const files = folder.getFilesByType(MimeType.CSV);

  let sheet = spreadsheet.getSheetByName(OUTPUT_SHEET_NAME);
```

```
  if (!sheet) {
    sheet = spreadsheet.insertSheet(OUTPUT_SHEET_NAME);
  } else {
    // Limpa a aba existente para evitar duplicação de dados em execuções
subsequentes
    sheet.clearContents();
  }

  const allData = []; // Array para armazenar todos os dados
temporariamente
  const headers = []; // Array para armazenar os nomes dos arquivos
(cabeçalhos das colunas)
  let maxRows = 0; // Para garantir que todas as colunas tenham o mesmo
número de linhas no final

  // Coleta os dados de cada arquivo CSV
  while (files.hasNext()) {
    const file = files.next();
    const fileName = file.getName();
    const fileContent = file.getBlob().getDataAsString();

    headers.push(fileName); // Adiciona o nome do arquivo como cabeçalho

    const columnData = []; // Para armazenar a segunda coluna deste arquivo

    const lines = fileContent.split(/\r\n|\n/);

    for (let i = 0; i < lines.length; i++) {
      const line = lines[i];
      if (line.trim() === '') continue; // Ignora linhas vazias

      // Divide a linha em colunas. Adapte se seu CSV usa outro
delimitador.
      const columns = line.split(',');

      if (columns.length >= 2) {
        columnData.push(columns[1]); // Pega apenas a segunda coluna
      } else {
```

```
        Logger.log(`Arquivo ${fileName}: Linha ${i + 1} não tem segunda
coluna: "${line}"`);
      }
    }
    allData.push(columnData);
    if (columnData.length > maxRows) {
      maxRows = columnData.length;
    }
    Logger.log(`Dados da segunda coluna coletados de: ${fileName}`);
  }

  // Prepara os dados para escrita na planilha
  // Agora transformamos o array de colunas em um array de linhas,
preenchendo com vazios se necessário
  const finalData = [];

  // Adiciona a linha de cabeçalho
  finalData.push(headers);

  // Preenche o restante das linhas
  for (let r = 0; r < maxRows; r++) {
    const row = [];
    for (let c = 0; c < allData.length; c++) {
      // Pega o dado se existir, caso contrário, usa vazio
      row.push(allData[c][r] !== undefined ? allData[c][r] : '');
    }
    finalData.push(row);
  }

  // Escreve todos os dados de uma vez na planilha, o que é muito mais
eficiente
  if (finalData.length > 0 && finalData[0].length > 0) {
    sheet.getRange(1, 1, finalData.length,
finalData[0].length).setValues(finalData);
  } else {
    Logger.log('Nenhum dado encontrado para consolidação.');
  }
```

```
  Logger.log('Consolidação por coluna concluída na planilha: ' +
spreadsheet.getUrl());
}
```