Universidade Estadual Paulista "Júlio de Mesquita Filho"
Instituto de Biociências, Letras e Ciências Exatas - Campus de São José do Rio Preto

JÚLIA RODRIGUES MARQUES DO NASCIMENTO

Análise Comparativa entre o Algoritmo Clássico e Quântico de K-Vizinhos Mais Próximos na Classificação de Vocalizações de Aves :

Aplicações de *Machine Learning* na Bioacústica com Enfoque em Computação Quântica

São José do Rio Preto
2025

Júlia Rodrigues Marques do Nascimento

Análise Comparativa entre o Algoritmo Clássico e Quântico de K-Vizinhos Mais Próximos na Classificação de Vocalizações de Aves :

Aplicações de *Machine Learning* na Bioacústica com Enfoque em Computação Quântica

Trabalho de Conclusão de Curso, apresentada à Universidade Estadual Paulista (UNESP), Instituto de Biociências, Letras e Ciências Exatas , São José do Rio Preto, para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Profº Dr. Eng. Rodrigo Capobianco Guido

São José do Rio Preto
2025

IMPACTO POTENCIAL DESTA PESQUISA

Esta pesquisa apresenta impacto potencial em duas vertentes principais: a computacional e a ambiental.

No âmbito computacional, contribui ao investigar formas de aprimorar a capacidade de classificação de algoritmos de Aprendizado de Máquina (*Machine Learning* - ML) por meio da combinação de técnicas de pré-processamento e extração de características. Desse modo, o estudo busca utilizar algoritmos mais leves computacionalmente, como o K-Vizinhos Mais Próximos (KNN), demonstrando que, em muitos casos, este algoritmo é capaz de alcançar desempenho comparável ao de classificadores mais robustos, como Redes Neurais. Ademais, incorpora a Computação Quântica, área emergente e promissora para a resolução de problemas complexos, mas que ainda está em fase inicial de estudo e possui bastante espaço para novas pesquisas.

No âmbito ambiental, o trabalho concentra-se na identificação de sons de aves brasileiras como ferramenta para o monitoramento da biodiversidade, utilizando as técnicas computacionais supracitadas para classificar as espécies por meio de suas vocalizações. Esta abordagem permite acompanhar alterações nos ecossistemas e contribui para a conservação da fauna, considerando o contexto dos impactos ambientais decorrentes de atividades antrópicas, como o desmatamento, por exemplo.

POTENTIAL IMPACT OF THIS RESEARCH

This research has potential impact in two main areas: computational and environmental.

In the computational domain, it contributes by investigating ways to improve the classification capacity of Machine Learning (ML) algorithms through the combination of preprocessing and feature extraction techniques. In this way, the study aims to use computationally lighter algorithms, such as K-Nearest Neighbors (KNN), demonstrating that, in many cases, this algorithm can achieve performance comparable to that of more robust classifiers, such as Neural Networks. Moreover, it incorporates Quantum Computing, an emerging and promising field for solving complex problems, which is still in the early stages of study and has considerable room for further research.

In the environmental domain, the work focuses on analyzing the sounds of Brazilian birds as a tool for biodiversity monitoring, using the aforementioned computational techniques to classify species based on their vocalizations. This approach allows for tracking changes in ecosystems and contributes to wildlife conservation, considering the context of environmental impacts resulting from anthropogenic activities, such as deforestation, for exemple.

JÚLIA RODRIGUES MARQUES DO NASCIMENTO

ANÁLISE COMPARATIVA ENTRE O ALGORITMO CLÁSSICO E QUÂNTICO DE K-VIZINHOS MAIS PRÓXIMOS NA CLASSIFICAÇÃO DE VOCALIZAÇÕES DE AVES :

Aplicações de *Machine Learning* na Bioacústica com Enfoque em Computação Quântica

Trabalho de Conclusão de Curso apresentado à Universidade Estadual Paulista (UNESP), Instituto de Biociências, Letras e Ciências Exatas , São José do Rio Preto, para obtenção do título de Bacharel em Ciência da Computação.

Data de defesa: 19/11/2025

BANCA EXAMINADORA

_____

Profº Dr. Eng. Rodrigo Capobianco Guido
UNESP – Instituto de Biociências, Letras e Ciências Exatas

_____

Profº Dra. Renata Spolon Lobato
UNESP – Instituto de Biociências, Letras e Ciências Exatas

_____

Profº Dr. Diego Renan Bruno
UNESP – Instituto de Biociências, Letras e Ciências Exatas

Dedico este trabalho ao meio ambiente, na esperança de que mais pessoas busquem formas de protegê-lo e que todos os seres possam viver em harmonia.

AGRADECIMENTOS

Primeiramente, agradeço aos meus pais: à minha mãe, Luciana Vieira de Novaes Rodrigues, por me incentivar a seguir a carreira em computação e por todo o apoio ao longo desta jornada; e ao meu pai, Milton Marques do Nascimento, pelo suporte durante esse período.

Agradeço aos demais familiares, aos meus amigos e ao meu namorado pelo encorajamento e carinho ao longo da minha trajetória acadêmica. De forma especial, registro minha gratidão ao meu tio, Mário Alexandre Zocal, cujo incentivo foi muito importante para o meu ingresso e continuidade no ensino superior.

Gostaria de agradecer também à minha gatinha Nina, que esteve ao meu lado ao longo do curso e me ajudou a lidar com os momentos difíceis.

Ao meu orientador, Professor Dr. Eng. Rodrigo Capobianco Guido, expresso meus agradecimentos por todas as oportunidades, pelos ensinamentos e pela dedicação, sempre me auxiliando quando necessário na elaboração deste trabalho.

Agradeço igualmente a Joji Suda, pelo suporte no decorrer do estudo em Computação Quântica, e aos Professores Dr. Wallace Oliveira Casaca e Dra. Marilaine Colnago, que contribuíram para o início da minha trajetória científica durante os primeiros anos do ensino superior.

Por fim, deixo meus agradecimentos a todos os professores que me acompanharam no decorrer do curso, contribuindo significativamente para minha formação acadêmica, profissional e pessoal.

“What you do makes a difference, and you have to decide what kind of difference you want to make.“

(GOODALL; BERMAN, 1999)

RESUMO

As aves desempenham um papel ambiental fundamental, estando presentes em diversos nichos e habitats devido à sua grande adaptabilidade a diferentes meios. Dessa forma, esses animais funcionam como importantes bioindicadores, fornecendo informações sobre a saúde dos ecossistemas por meio de parâmetros como desaparecimento, redução populacional, padrões de movimentação, entre outros. Com o aumento do aquecimento global e de atividades humanas que geram devastação e poluição, torna-se essencial compreender o estado do meio ambiente para subsidiar a tomada de decisões futuras. Nesse contexto, a tecnologia apresenta-se como ferramenta estratégica para monitoramento e proteção ambiental, permitindo a extração de informações relevantes do ambiente. Assim, este trabalho investiga a aplicação do algoritmo KNN no monitoramento de 20 espécies brasileiras de aves, avaliando seu desempenho na classificação multi-classe, de modo que este número possa ser ampliado conforme a performance do algoritmo. Considerando que redes neurais demandam maior poder computacional, optou-se por explorar um algoritmo simples de ML, aprimorado por técnicas de pré-processamento e extração de características. Além disso, o estudo contempla a versão quântica do KNN, evidenciando o potencial emergente desse tipo de tecnologia para aplicações de classificação. Os resultados obtidos indicam acurácia de aproximadamente 98,66% para o KNN clássico e de cerca de 89% para a versão quântica no melhor caso de ambos, como será apresentado no trabalho, demonstrando que abordagens leves podem ser eficazes em aplicações de bioacústica.

Palavras-Chave: Processamento Digital de Sinais (PDS); Computação Quântica; *Machine Learning* (ML); Biodiversidade; K-Vizinhos Mais Próximos (KNN); Extração de Características.

ABSTRACT

Birds play a fundamental environmental role, occupying various niches and habitats due to their high adaptability to different environments. In this way, these animals serve as important bioindicators, providing information about ecosystem health through parameters such as disappearance, population decline, movement patterns, among others. With the increase in global warming and human activities that cause devastation and pollution, understanding the state of the environment becomes essential to support future decision-making. In this context, technology emerges as a strategic tool for environmental monitoring and protection, allowing the extraction of relevant information from the environment. Thus, this work investigates the application of the KNN algorithm in monitoring 20 Brazilian bird species, evaluating its performance in this multi-class classification, with the possibility of expanding this number depending on the algorithm's effectiveness. Considering that neural networks require greater computational power, a simple ML algorithm was chosen, enhanced through preprocessing and feature extraction techniques. Moreover, the study also considers the quantum version of KNN, highlighting the emerging potential of this type of technology for classification applications. The results obtained indicate an accuracy of approximately 98,66% for classical KNN and approximately 89% for the quantum version in the best case for both algorithms, as will be presented in the study, demonstrating that lightweight approaches can be effective in bioacustic applications.

Keywords: Signal Digital Processing (SDP); Quantum Computing; Machine Learning; Biodiversity; K-Nearest Neighboors (KNN); Feature Extraction.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

DC	Corrente Contínua (*Direct Current*)
DCT	Transformada Discreta do Cosseno (*Discrete Cosine Transform*)
DFT	Transformada Discreta de Fourier (*Discrete Fourier Transform*)
FFT	Transformada Rápida de Fourier (*Fast Fourier Transform*)
HQC	Híbrido Quântico-Clássico (*Hybrid Quantum-Classical*)
IA	Inteligência Artificial
IDE	Ambiente Integrado de Desenvolvimento (*Integrated Development Environment*)
KNN	K-Vizinhos Mais Próximos (*K-Nearest Neighbors*)
MFCC	Coeficientes Cepstrais em Frequências Mel-Escalonadas (*Mel-Frequency Cepstral Coefficients*)
ML	Aprendizado de Máquina (*Machine Learning*)
NISQ	Dispositivos Quânticos Ruidosos de Escala Intermediária (*Noisy Intermediate-Scale Quantum*)
PCA	Análise de Componentes Principais (*Principal Component Analysis*)
PDS	Processamento de Sinais Digitais
PR	Reconhecimento de Padrões (*Pattern Recognition*)
QKNN	K-Vizinhos Mais Próximos Quântico (*Quantum K-Nearest Neighbors*)
QML	Aprendizado de Máquina Quântico (*Quantum Machine Learning*)
TEO	Operador de Energia de *Teager* (*Teager Energy Operator*)
TKEO	Operador de Energia de *Teager-Kaiser* (*Teager-Kaiser Energy Operator*)
ZCR	Taxa de Cruzamento por Zero (*Zero Crossing Rate*)

LISTA DE SÍMBOLOS

α	Letra grega Alfa
β	Letra grega Beta
γ	Letra grega Gama
δ	Letra grega Delta
λ	Letra grega Lambda
μ	Letra grega Mi
π	Letra grega Pi
σ	Letra grega Sigma
θ	Letra grega Teta
ϕ	Letra grega Phi
φ	Letra grega Phi variante
ψ	Letra grega Psi
$\log$	Função logaritmo
$\sin$	Função seno
$\cos$	Função cosseno
e	Número de Euler
$\sum$	Operador de somatório
$\otimes$	Símbolo de produto tensorial
$\mathbb{R}$	Conjunto dos números reais
$\mathbb{Z}$	Conjunto dos números inteiros

SUMÁRIO

1 INTRODUÇÃO

1.1 CONSIDERAÇÕES INICIAIS

Nos últimos anos, tem-se observado a intensificação de desastres ambientais, muitos dos quais estão diretamente relacionados às ações antrópicas, como o desmatamento, a emissão de gases de efeito estufa, dentre outras práticas nocivas ao meio ambiente. Esses fatores têm contribuído para a ocorrência de eventos climáticos extremos, afetando negativamente a economia, a agricultura, a saúde e a qualidade de vida da população.

Em 2024, o Brasil voltou a liderar na questão da perda de floresta tropical mundial, respondendo por cerca de 42% de toda a área de vegetação primária tropical perdida globalmente naquele ano, durante uma das piores temporadas de queimadas e secas registradas nas últimas sete décadas (WRI, 2025). A fumaça alcançou grandes cidades, incluindo São José do Rio Preto, provocando problemas respiratórios na população, alterações térmicas, redução da luminosidade, entre outros efeitos. Além disso, a situação foi agravada pelo fenômeno El Niño, caracterizado pelo aquecimento anômalo das águas do Oceano Pacífico, que altera padrões de chuva e circulação de massas de ar (INMET, 2025). Embora natural, o El Niño tem se tornado mais intenso e irregular devido às mudanças climáticas globais (WMO, 2024), intensificando desastres ambientais.

Diante desse cenário, destaca-se também a redução da biodiversidade, que compromete o equilíbrio dos ecossistemas, uma vez que cada organismo desempenha funções essenciais no seu habitat, como polinização, controle populacional de outros indivíduos e ciclagem de nutrientes. Esse desequilíbrio aumenta a vulnerabilidade ambiental e gera consequências diretas para a sociedade, incluindo menor produção de alimentos pela falta de polinizadores, escassez de recursos ambientais e maior exposição a desastres naturais. Nesse sentido, o monitoramento da biodiversidade de uma região permite avaliar a saúde ecossistêmica, identificar alterações no meio e subsidiar a implementação de medidas de conservação.

Nesse contexto, as aves estão entre os indivíduos que mais rapidamente respondem às mudanças ambientais, sendo indicadores importantes sobre a saúde dos ecossistemas. Porém, também estão sendo ameaçadas mediante as mudanças climáticas, poluição e interferências no meio ambiente. Segundo dados do *The Guardian*, publicados em outubro de 2025, 61% das espécies de aves do planeta estão em declínio, o que denuncia a gravidade do impacto ambiental resultante das ações antrópicas (GREENFIELD, 2025).

Paralelamente a esses fatores, pesquisas e investimentos em novas tecnologias têm aumentado, principalmente na área da computação. Em 2024, os gastos globais associados ao mercado de *software* atingiram $ 899,9 bilhões, representando um aumento de quase 11,9% em relação ao ano anterior (GR, 2024), evidenciando o rápido crescimento dessas ferramentas. Embora o uso inadequado da tecnologia possa agravar problemas ambientais, ela também pode ser empregada

para mitigá-los, auxiliando no monitoramento de ecossistemas e servindo como instrumento para mecanismos de conservação.

A Inteligência Artificial (IA) é um dos campos da computação que mais têm se expandido. Segundo a pesquisa da McKinsey (MCKINSEY, 2024), a adoção de IA em empresas alcançou 72% em 2024, contra 55% em 2023, evidenciando seu potencial na automatização de tarefas. A IA permite reduzir erros humanos em atividades repetitivas, aumentando a velocidade e qualidade de execução. Dessa forma, ela pode ser aplicada para fins ambientais, como exemplificado no trabalho "*A Robust Dual-Mode Machine Learning Framework for Classifying Deforestation Patterns in Amazon Native Lands*", em que foi usado IA para identificar áreas desmatadas da Amazônia por meio de imagens de Sensoriamento Remoto (SR), dentro do território do São Félix do Xingu (PA) (RODRIGUES et al., 2024).

Concomitantemente, outra tecnologia em avanço é a Computação Quântica, que oferece potencial para reduzir o tempo de processamento, aumentar a capacidade de armazenamento e ampliar o poder computacional. *Quantum Machine Learning* (QML) já foi aplicado em contextos ambientais, especialmente em tarefas relacionadas a mudanças climáticas e sustentabilidade, e se destacaram por terem processado conjuntos de dados complexos com maior eficiência do que métodos clássicos em alguns casos (NAMMOUCHI; KASSLER; THEORACHIS, 2025). Logo, explorar a aptidão da Computação Quântica para fins de bio-monitoramento é uma estratégia positiva, já que o monitoramento de espécies depende de grandes quantidades de informações complexas.

Dessa forma, esta pesquisa pretende explorar o uso de algoritmos clássicos e quânticos do KNN para identificar 20 espécies brasileiras de aves, comparando ambos os paradigmas na resolução do problema proposto dentro da bioacústica.

1.2 JUSTIFICATIVA

Como discutido, as atividades humanas têm impactado negativamente o meio ambiente. Entre esses impactos, a redução da biodiversidade acelera mudanças nos ecossistemas (HOPER et al., 2012), causando desequilíbrios no meio.

Neste contexto, as aves se destacam como bioindicadores eficientes (MEKONEN, 2017), capazes de refletir rapidamente alterações ambientais. Isso ocorre porque ocupam diferentes nichos ecológicos, apresentam posições variadas na cadeia alimentar e possuem padrões de migração e reprodução diretamente afetados pelas mudanças climáticas. Entretanto, o monitoramento de espécies de aves é desafiador, tornando a classificação automática por IA uma ferramenta promissora para apoiar as observações.

Logo, dada a sua relevância ambiental, as aves foram escolhidas como foco deste estudo. Considerando que o Brasil abriga grande diversidade de espécies, sendo muitas delas endêmicas (PACHECO et al., 2021), o país foi selecionado como área de investigação.

Outrossim, como muitas espécies de aves camuflam-se no ambiente, optou-se por utilizar

suas vocalizações para a classificação, por meio de Processamento Digital de Sinais (PDS), uma abordagem menos invasiva e muito usada por facilitar o monitoramento em áreas extensas ou de difícil acesso.

Para a aplicação de classificadores automatizados, optou-se pelo algoritmo K-Vizinhos Mais Próximos (KNN) por sua simplicidade e baixo custo computacional. Assim, buscou-se avaliar se modelos leves podem alcançar desempenho competitivo frente a modelos mais complexos, como redes neurais. Além disso, essa abordagem possibilitou analisar de forma mais transparente o impacto do pré-processamento e da extração de características no desempenho final da classificação.

Por fim, a Computação Quântica foi escolhida por ser um paradigma emergente, com potencial promissor para acelerar tarefas de aprendizado de máquina (ML) e aprimorar a inferência em conjuntos complexos de dados. Considerando que se trata de uma área relativamente recente, com inúmeras possibilidades de pesquisa, este trabalho também buscou investigar seu comportamento especificamente na tarefa de bioacústica, verificando se traz vantagens expressivas no ramo de PDS.

1.3 OBJETIVOS

Mediante o que foi discutido, o principal objetivo desta pesquisa é desenvolver e aplicar o algoritmo de KNN na tarefa de classificação e reconhecimento de 20 espécies de aves brasileiras a partir de suas vocalizações, comparando o paradigma clássico e quântico do classificador.

Ademais, este trabalho busca explorar diferentes estratégias de pré-processamento e de polimento dos dados, investigando como a combinação desses fatores impacta no desempenho final dos modelos.

Em relação aos objetivos específicos, esta pesquisa visa:

1. Abordar um problema ambiental relevante e incentivar pesquisas que integrem tecnologia com conservação do meio ambiente.

2. Investigar estratégias de aprimoramento de algoritmos leves de classificação (como o KNN) por meio de pré-processamento de dados, extração de características e ajuste de hiperparâmetros, contribuindo para as áreas de PDS e Inteligência Artificial.

3. Explorar o potencial de classificadores quânticos, que ainda são limitados, incentivando o avanço de pesquisas e aplicações na área de Computação Quântica.

4. Desenvolver um modelo de classificação de espécies capaz de ser escalonado.

5. Comparar métricas de desempenho entre as versões clássica e quântica do KNN, analisando suas diferenças e limitações.

1.3.1 Organização do trabalho

O texto deste trabalho está dividido da seguinte maneira:

- O Capítulo 2 apresenta o referencial teórico, contextualizando os métodos utilizados.
- O Capítulo 3 apresenta os trabalhos anteriores relacionados ao tema, que serviram de inspiração para esta pesquisa.
- O Capítulo 4 detalha a metodologia e o processo de implementação do que foi explicado no Capítulo 2.
- O Capítulo 5 expõe e analisa os resultados obtidos mediante métricas estatísticas.
- O Capítulo 6 apresenta as conclusões e propostas para trabalhos futuros.

2 REFERENCIAL TEÓRICO

Neste capítulo, apresenta-se a fundamentação teórica que embasa este trabalho, abordando os principais conceitos, métodos e técnicas implementados, de modo a fornecer a base necessária para a compreensão da metodologia e dos resultados.

2.1 BIOACÚSTICA

A bioacústica é um campo interdisciplinar da biologia e acústica, estudando os sons dos animais e os mecanismos de comunicação entre os indivíduos de uma mesma espécie (TUBARO, 1999). Essa área ganhou destaque a partir da segunda metade do século XX, quando ferramentas de gravação e armazenamento de áudio se popularizaram, expandindo-se ainda mais devido ao maior acesso a equipamentos de computação nas últimas décadas.

Boa parte dos estudos em bioacústica estão concentrados em pesquisas sobre etologia (comportamento dos animais) e sistemática (classificação de espécies). Ademais, diferentes estudos voltados à preservação, conservação e manejo de espécies naturais utilizam a bioacústica como ferramenta para monitoramento da biodiversidade e avaliação de interações ambientais (BAPTISTA; GAUNT, 1997).

Uma das principais vantagens do uso de bioacústica para estudos do meio ambiente é o seu caráter não invasivo, possibilitando coleta contínua de dados em tempo real e o monitoramento em áreas de difícil acesso (GIBB et al., 2018).

Tecnologias como Processamento Digital de Sinais (PDS) e Inteligência Artificial (IA) têm ampliado a bioacústica, permitindo automatizar a coleta e análise de dados ambientais, tarefa que seria longa e sujeita a erros se feita manualmente (SHARMA; SATO; GAUTAM, 2023). No entanto, ruídos e a complexidade da identificação de espécies ainda representam desafios, tornando o uso de aprendizado de máquina associado a PDS essencial para extração de características relevantes e classificação automatizada dos sons (MUTANU et al., 2022).

2.2 PROCESSAMENTO DIGITAL DE SINAIS

Processamento Digital de Sinais (PDS) consiste em um conjunto de técnicas para representação, manipulação e transformação da informação contida em um sinal (OPPENHEIM; SCHAFER, 2010). Um sinal é uma função constituída por uma ou mais variáveis representando mudanças em uma entidade quantificável. Logo, para ser considerado um sinal, é necessário haver alguma variação observável na função ao longo do tempo ou do espaço (CHAKRAVORTY, 2018).

Existem dois tipos de sinais: os contínuos e os discretos. Sinais contínuos correspondem a funções, como $u(t)$, que estão definidas para todo valor da variável independente, como t no

caso (MENDEZ, 2023).

O segundo tipo é o sinal discreto, que engloba funções cujos valores só existem em pontos específicos da variável independente. Por exemplo, pode-se haver uma função $u(k)$, em que k exite apenas em pontos selecionados do tempo ou espaço (MENDEZ, 2023).

Os sinais possuem diferentes usos, estando presentes em medicina (imagens e áudios médicos), geografia (imagens de sensoriamento remoto), biologia (sons de espécies), entre outras áreas, caracterizando PDS como um campo com atuação interdisciplinar.

2.3 RECONHECIMENTO DE PADRÕES (PR)

Em 1955, Selfridge introduziu o conceito de reconhecimento de padrões (PR), definindo-o como a extração de características relevantes a respeito de uma classes partir de um conjunto de dados, separando-as de informações irrelevantes (SELFRIDGE, 1955). Em geral, utiliza-se PR para classificar amostras mediante os padrões que elas apresentam, identificados por propriedades significativas que permitam a separação entre os grupos.

Matematicamente, um padrão descreve um objeto por meio de atributos compartilhados e pode ser representado como um vetor de características, conforme a Equação 1, na qual cada x_i representa um atributo ou característica do objeto.

$$\mathbf{x} = [x_1, x_2, \ldots, x_n] \tag{1}$$

Já uma classe $\mathcal{C}_k$ é definida como um conjunto de padrões semelhantes, como representado na Equação 2.

$$\mathcal{C}_k = \{\mathbf{x} \mid \mathbf{x} \text{ pertence à classe } k\} \tag{2}$$

Há ainda um vetor característico $\mathbf{V}$, que reúne as propriedades que distinguem os padrões de cada classe, selecionando apenas os atributos relevantes para a separação entre elas.

2.4 EXTRAÇÃO DE CARACTERÍSTICAS NO RECONHECIMENTO DE PADRÕES

O sinal bruto possui grande quantidade de informações, incluindo elementos redundantes e prejudiciais à classificação. Por isso, realiza-se a etapa de pré-processamento e, em seguida, aplica-se a extração de características, que transforma os dados originais em um conjunto reduzido de atributos representativos da classe correspondente, como explicado na seção sobre PR.

Na literatura, encontra-se duas abordagens principais de extração de características: a manual, quando os atributos são selecionados com base no conhecimento prévio sobre os dados; e a automática (*Feature Learning*), em que os próprios modelos de aprendizado profundo aprendem os atributos relevantes diretamente a partir das amostras (SHI et al., 2025). Neste trabalho,

optou-se pela extração manual de características, a fim de manter maior controle sobre o fluxo do processamento do sinal.

2.4.1 Taxa de Cruzamento por Zero (ZCR)

A Taxa de Cruzamento por Zero (*Zero Crossing Rate* – ZCR) é um método de extração de características simples e computacionalmente leve, uma vez que extrai informações dos sinais no domínio do tempo, sem a necessidade de conversão explícita para o domínio da frequência (GUIDO, 2016b).

O ZCR consiste em calcular a quantidade de vezes em que um sinal em formato de onda cruza a amplitude zero (GUIDO, 2016b). Matematicamente, considerando $s[\cdot] = \{s_0, s_1, s_2, \ldots, s_{M-1}\}$ um sinal discreto de comprimento $M > 1$, o ZCR é definido pela Equação 3, sendo a função sinal (*sign*) apresentada na Equação 4.

$$\text{ZCR}(s[\cdot]) = \frac{1}{2} \sum_{j=0}^{M-2} |\,\text{sign}(s_j) - \text{sign}(s_{j+1})| \tag{3}$$

$$\text{sign}(x) = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases} \tag{4}$$

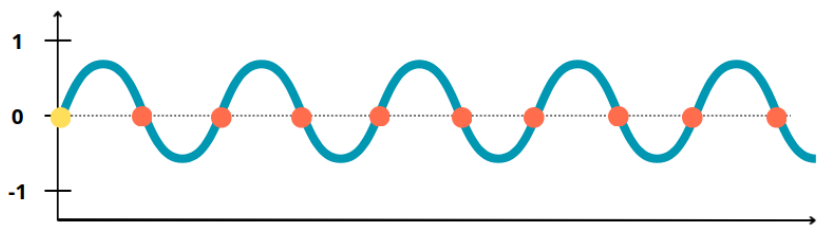
Note que $\text{ZCR}(s[\cdot]) \geq 0$ para qualquer $s[\cdot]$, não assumindo valores negativos.

Valores altos de ZCR indicam que o sinal muda rapidamente de quadrante, associado a componentes de alta frequência. Valores baixos de ZCR indicam trechos mais estáveis, com predominância de baixas frequências.

A desvantagem deste método é sua sensibilidade a ruídos e variações de fundo no áudio, que são muito presentes em gravações ambientais. Por isso, ele pode ser combinado com outras técnicas, utilizando-o como auxiliar para garantir análises mais robustas na classificação.

Na Figura 1, ilustra-se o ZCR. O sinal é representado como uma onda em um gráfico, variando sua amplitude entre -1 e 1. Os pontos laranja indicam os momentos em que o sinal cruza o eixo zero em y, totalizando 9 cruzamentos. O ponto amarelo marca o início do sinal, sem indicar um cruzamento naquele instante. No gráfico, o eixo x representa o tempo e o eixo y a amplitude do sinal.

Figura 1 – Exemplo de sinal para explicação do ZCR.



Fonte: Produzido pelos autores.

2.4.2 Operador de Energia de *Teager* (TEO)/*Teager–Kaiser* (TKEO)

O TEO, ou TKEO, é um operador matemático não linear, aplicado a um sinal para estimar sua energia instantânea (KAISER, 1990). Por ser não linear, sua saída não é diretamente proporcional à entrada, permitindo capturar variações sutis de energia associadas às mudanças de amplitude e frequência do sinal.

A função aplicada em sinais contínuos é representada por ψ, com a fórmula na Equação 5.

$$\psi(x(t)) = x'^2(t) - x(t)x''(t) \tag{5}$$

Resumidamente, $x(t)$ representa o sinal contínuo observado no tempo t, $x'(t)$ sua primeira derivada e $x''(t)$ sua segunda derivada. A primeira derivada representa a taxa de variação da amplitude ao longo do tempo, ou seja, a velocidade com que o sinal se altera. Já a segunda derivada representa a variação da primeira, o que seria a aceleração dessa variação identificada no sinal.

Entretanto, para ser analisado pelo computador, o sinal deve ser digitalizado, por meio de amostragem e quantização, passando a ser representado na forma discreta, com uma pequena perda de informação. Nesse caso, o o TEO/TKEO é expresso pela Equação 6, usando a amostra atual ($x(n)$), a anterior e a posterior ($n-1$ e $n+1$, respectivamente).

$$\psi[x(n)] = x^2(n) - x(n+1)x(n-1) \tag{6}$$

Dessa forma, o operador integra informações de amplitude e frequência, fornecendo uma estimativa da energia dinâmica do sinal em cada instante (PINHEIRO et al., 2025), o que o torna adequado para a análise de sinais não estacionários (com mudanças abruptas) e de curta duração, como os utilizados neste estudo.

Apesar de ser eficaz, o TEO/TKEO é sensível ruídos, podendo ser combinado com outros métodos para melhor desempenho, além de requirir em alguns casos filtragem ou suavização do sinal.

2.4.3 Método baseado em energia acumulada

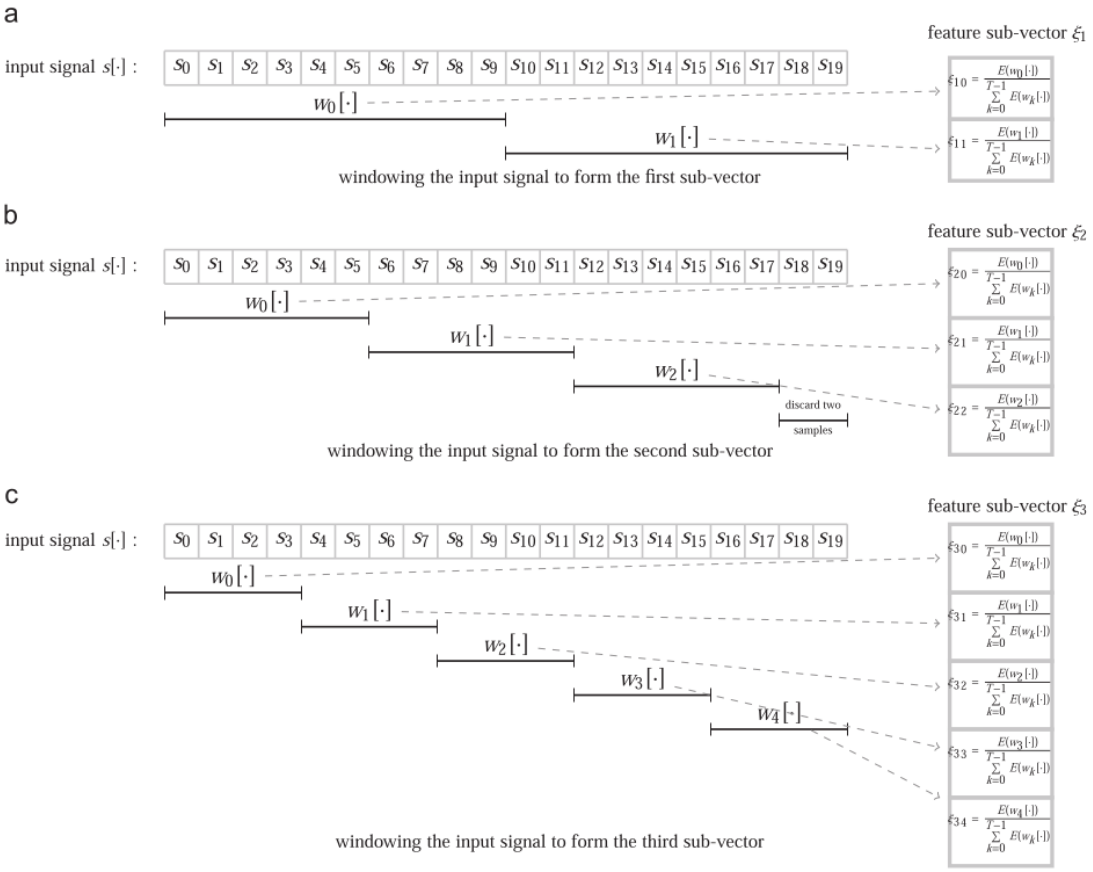
Um método de extração manual de características utilizado foi o de energia acumulada, que calcula a soma dos quadrados das amplitudes do sinal ao longo do tempo, refletindo a quantidade total de energia contida no segmento analisado (GUIDO, 2016a). Para sinais unidimensionais, como os de áudio, a energia E pode ser definida como na Equação 7, na qual x_i representa a i-ésima amostra do sinal com N total de amostras. Logo, em um sinal, a energia é a medida de sua intensidade ou magnitude ao longo do tempo ou espaço.

$$E = \sum_{i=0}^{N-1} x_i^2 \tag{7}$$

O método de energia aplicado durante a realização deste trabalho foi retirado do artigo "*A tutorial on signal energy and its applications*", de (GUIDO, 2016a). Nesse estudo, Guido (2016a) apresentou três maneiras de extração de características por energia em sinais digitais, sendo o A3 o único que determina o tamanho do janelamento, ou seja, define o tamanho das divisões no sinal para alcançar um determinado nível de energia acumulada, sem a necessidade de janelas de tamanho fixo. Dessa forma, é possível medir progressivamente a fração do sinal digital necessária para atingir um certo nível energético.

No procedimento adotado (A3), o sinal é dividido em sub-janelas w_i e define-se um limiar de energia C ($0 < C < 100$). Para cada sub-janela, calcula-se a fração do sinal necessária para atingir $i \cdot C\%$ da energia total, formando o vetor de características $f = [f_0, f_1, \ldots, f_{T-1}]$, onde f_0 corresponde à proporção do sinal necessária para atingir $C\%$ da energia, f_1 a $2C\%$, e assim sucessivamente até $T \cdot C = 100\%$. Dessa forma, o método codifica como a energia do sinal se acumula ao longo do tempo, permitindo observar diretamente trechos mais ou menos intensos. Na Figura 2, ilustra-se o procedimento com diferentes granularidades em (a), (b) e (c); janelas sucessivas são usadas para calcular a razão entre a energia acumulada e a energia total, gerando um vetor que representa a estrutura energética do sinal.

Figura 2 – Exemplo de extração de características por energia.



Fonte: (GUIDO, 2016a)

2.4.4 Coeficientes Cepstrais da Frequência Mel (MFCC)

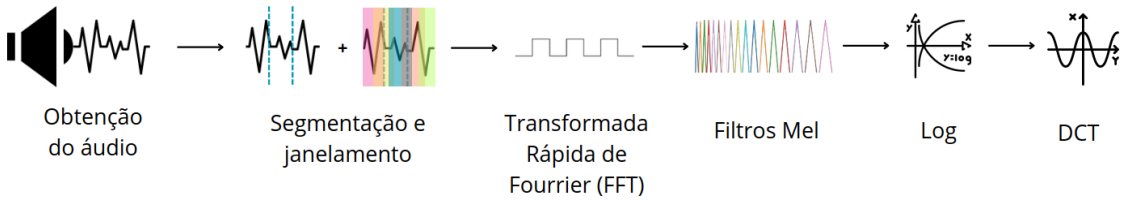
O MFCC é um procedimento de PDS amplamente usado, representando o espectro de potência de um som, ou seja, como a energia do sinal se distribui entre diferentes frequências ao longo do tempo. Para isso, utiliza a escala Mel, que é melhor em representar os aspectos do espectro da fala (DAVIS; MERMELSTEIN, 1980), sendo mais próxima da percepção humana desses sinais. Devido essas propriedades, a técnica é muito usada em reconhecimento de padrões em sons de animais, identificação de falas, entre outras aplicações.

De acordo com Gupta (2013) os passos para calcular o MFCC de um sinal 1D (áudio) são (GUPTA et al., 2013):

1. Segmentação do sinal em blocos (*blocking*) ou quadros (*framing*);
2. Aplicação de janelamento em cada quadro;
3. Cálculo da Transformada Rápida de Fourier (FFT) para obter o espectro de potência;
4. Aplicação de filtros triangulares na escala Mel;
5. Cálculo do logaritmo da energia em cada faixa filtrada;
6. Aplicação da Transformada Discreta do Cosseno (DCT) para obter os coeficientes MFCC;

Na Figura 3, apresenta-se o fluxograma das etapas supracitadas.

Figura 3 – Etapas do MFCC.



Fonte: produzido pelos autores.

2.4.4.1 Segmentação do sinal

Primeiramente, o sinal é dividido em quadros/blocos de N amostras, com um certo nível de sobreposição. Assim, cada novo quadro começa M amostras após o anterior, com $M < N$, ocasionando uma sobreposição de $N - M$ quadros, isso para suavizar as transições e impedir a perda de informação. Os valores padrões, que costumam ser usados por gerarem bons resultados e que foram escolhidos neste trabalho, são: $N = 256$ e $M = 100$.

2.4.4.2 Janelamento

Na segunda etapa, aplica-se o janelamento para minimizar descontinuidades nas bordas de cada quadro, oriundas da segmentação. Cada quadro $X_n(m)$ é multiplicado por uma função de janelamento $W_n(m)$, com m sendo o índice da amostra e N_m o total de amostras no quadro, como representado na Equação 8.

$$Y_n(m) = X_n(m)\, W_n(m), \quad 0 \leq m \leq N_m - 1 \tag{8}$$

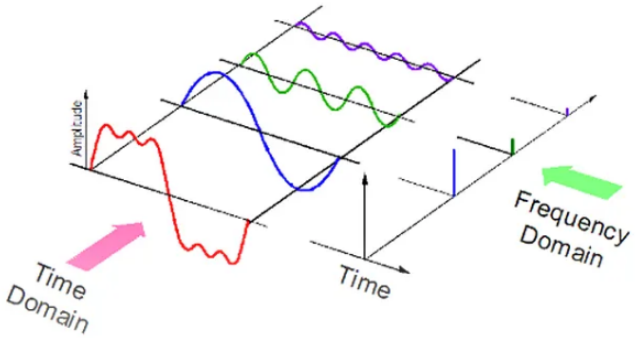
Na Equação 8, $Y_n(m)$ é o sinal resultante após o janelamento. Existem várias funções janela, como retangular, flat-top e Hamming, sendo a última a mais utilizada por reduzir o vazamento espectral, como apresentada pela Equação 9.

$$W_n(m) = 0,54 - 0,46 \cos\left(\frac{2\pi m}{N_m - 1}\right), \quad 0 \leq m \leq N_m - 1 \tag{9}$$

2.4.4.3 Transformada Rápida de Fourier (FFT)

A FFT converte um sinal do domínio temporal para o domínio da frequência. Na Figura 4, enquanto o domínio temporal mostra a forma da onda ao longo do tempo (em relação à sua amplitude), o domínio da frequência evidencia a magnitude de cada frequência, facilitando a compreensão da estrutura espectral e a aplicação de filtros.

Figura 4 – Diferença entre domínio de tempo e domínio da frequência.



Fonte: (ZAVALA, 2025).

A FFT é uma versão eficiente da Transformada Discreta de Fourier (DFT), pois atua em sub-blocos do sinal, reduzindo a complexidade computacional dos cálculos. Sua saída consiste em números complexos, mas no método utiliza-se apenas o valor absoluto desses números (magnitude) para as etapas seguintes.

A Transformada Discreta de Fourier (DFT) é definida pela Equação 10, na qual x_m são as amostras do quadro e X_k os coeficientes de frequência obtidos.

$$X_k = \sum_{m=0}^{N_m-1} x_m \, e^{-j2\pi km/N_m}, \quad k = 0, 1, \ldots, N_m - 1 \tag{10}$$

2.4.4.4 Filtros de Mel

Após a aplicação do FFT, o espectro da potência é obtido e mapeado para a escala Mel por meio de um banco de filtros triangulares passa-banda com espaçamento e largura definidos pela escala Mel. Tal mapeamento permite estimar a energia em cada faixa perceptual. A conversão da frequência f em Hertz para a frequência na escala Mel m é dada pela Equação 11.

$$m = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right) \tag{11}$$

Com as estimativas energéticas em cada filtro, calcula-se o logaritmo de cada, chamado espectro Mel, que será usado nas etapas seguintes para a extração dos coeficientes cepstrais.

2.4.4.5 Transformada Discreta do Cosseno (DCT)

Na última etapa, aplica-se a DCT ao vetor de log-energias para gerar os coeficientes cepstrais (MFCCs). Esses coeficientes representam uma forma compacta da estrutura espectral do sinal, aproximando a forma como o sistema auditivo humano percebe variações de frequência. Em outras palavras, os MFCCs são um conjunto de características que descrevem o envelope espectral do áudio de maneira eficiente, concentrando a maior parte da informação relevante nos primeiros coeficientes.

Desse modo, obtêm-se os coeficientes cepstrais, mas utiliza-se apenas os primeiros, pois os coeficientes de ordem superior contêm variações rápidas e menos relevantes.

A fórmula para cálculo do m-ésimo coeficiente cepstral C_m é dada pela Equação 12.

$$C_m = \sum_{k-1}^{k} \log(X_k) \cdot \cos\left[\frac{\pi m}{k}\left(k - \frac{1}{2}\right)\right], \quad m = 0, 1, \ldots, k - 1 \tag{12}$$

Normalmente, os primeiros 13 coeficientes $C_0, C_1, \ldots, C_{12}$ são utilizados para representar cada quadro de áudio, pois contêm as informações mais relevantes do espectro para tarefas como classificação ou reconhecimento de padrões sonoros.

2.4.5 Análise de Componente Principal (PCA)

PCA é uma técnica estatística utilizada para redução da dimensionalidade de dados, com perda mínima de informação (JOLLIFFE; CADIMA, 2016). Para isso, o método realiza a combinação linear das variáveis iniciais, gerando um conjunto novo e reduzido de variáveis, que são denominadas componentes principais (PCs). Essa construção é feita de modo que a variância nos dados seja maximizada e que os componentes sejam mutuamente não correlacionados (JACKSON, 1991).

As etapas matemáticas aplicadas no *pipeline* do PCA, como apresentado por (GÉRON, 2019) e (GRUS, 2015), são:

1. Centralização dos dados;
2. Cálculo da matriz de covariância;
3. Decomposição em autovalores e autovetores;
4. Ordenação e seleção dos principais componentes;
5. Projeção dos dados e interpretação.

2.4.5.1 Centralização dos dados

É a primeira etapa, consistindo em subtrair a média de cada variável (que, no contexto de extração de *features*, corresponde à característica observada), garantindo uma média final igual a zero para cada uma.

Matematicamente, dados $X = [x_{ij}]$ com $i = 1, \ldots, n$ observações (linhas) e $j = 1, \ldots, m$ variáveis (colunas), a centralização é aplicada mediante a Equação 13.

$$x_{ij}^{centralizada} = x_{ij} - \mu_j, \quad \text{onde} \quad \mu_j = \frac{1}{n} \sum_{i=1}^{n} x_{ij} \tag{13}$$

Assim, a análise passa a considerar apenas a variação nos dados, sem influência do valor absoluto das variáveis ao fazer a decomposição da matriz de covariância, ou seja, impede que escalas diferentes interfiram erroneamente sobe os dados.

2.4.5.2 Cálculo da matriz de covariância

A matriz de covariância é uma matriz quadrada e simétrica que representa como duas ou mais variáveis se relacionam entre si. Considerando duas variáveis diferentes, uma em i e outra em j, por exemplo, o elemento na posição $[i, j]$ indica a covariância entre esse par.

Quando se utiliza a matriz de correlação (sinônimo para covariância), a diagonal principal assume sempre o valor 1, pois uma variável é perfeitamente correlacionada consigo mesma. A simetria da matriz ocorre porque a ordem das variáveis não altera a forma como a variação entre elas é medida.

Na Equação 14, representa-se como a covariância entre variáveis é calculada.

$$\operatorname{cov}(x, y) = \frac{\sum_{i=1}^{N} (x_i - \bar{x})\,(y_i - \bar{y})}{N - 1} \tag{14}$$

Por meio dessa matriz, determinam-se as direções (Componentes Principais - PC) que capturam a maior variabilidade dos dados, permitindo reduzir a dimensionalidade do conjunto original sem perder informação relevante.

2.4.5.3 Decomposição em auto-valores e auto-vetores

Autovalores, representados por λ, indicam o quanto um vetor é esticado, comprimido ou invertido ao ser transformado por uma matriz A. Se um vetor não nulo v satisfaz $Av = \lambda v$, então v é um autovetor de A, e λ é o autovalor associado. Autovetores mantêm sua direção sob a transformação, sofrendo apenas alteração de magnitude.

Os autovetores da matriz de covariância definem as direções dos componentes principais, enquanto os autovalores indicam a variância de cada componente.

2.4.5.4 Ordenação e seleção dos principais componentes

Após a etapa anterior, os autovalores obtidos são ordenados em ordem decrescente. Como o autovetor é a direção no espaço de dados e o autovalor a variação de cada componente, é esperado que os primeiros componentes capturem a maior parte da informação.

Logo, seleção dos componentes a serem mantidos pode ser feita de diferentes formas (WILMOTT, 2019), como:

- Definindo um número fixo de componentes principais;
- Mantendo apenas os componentes cuja variância seja superior a um limiar pré-definido;
- Selecionando os primeiros componentes que, em conjunto, expliquem um percentual mínimo desejado da variância total (por exemplo, 90%).

Dessa forma, mantém-se as informações mais relevantes e elimina-se redundâncias.

2.5 NORMALIZAÇÃO

Normalização dos dados é uma técnica que consiste em transformar as escalas dos valores para que todos estejam dentro de um mesmo intervalo, evitando que variáveis com valores muito maiores assumam maior relevância indevida na classificação (AMORIM; CAVALCANTI; CRUZ, 2023).

Diferentes métodos de normalização impactam de forma distinta no desempenho do modelo, dependendo da natureza dos dados. Os primeiros métodos apresentados aqui foram retirados do artigo "*The choice of scaling technique matters for classification performance*", de (AMORIM; CAVALCANTI; CRUZ, 2023).

2.5.1 *Standard Scale*

É um método que implementa uma normalização *Z-score*, padronizando os valores pela subtração da média ($\bar{x}$) e divisão pelo desvio padrão (σ), como na Equação 15. Dessa forma, os dados normalizados passam a ter média zero e desvio padrão igual a 1.

$$x'_i = \frac{x_i - mean}{\sigma} \tag{15}$$

Sua vantagem é a normalização de atributos com valores negativos ou positivos; porém, é limitada na presença de *outliers*, ficando muito comprimida.

2.5.2 *Min-max Scaler*

Este método normaliza em relação ao eixo x, mantendo os valores dentro do intervalo $[0, 1]$, como mostrado na Equação 16. O fator de escala é o intervalo do atributo, e o termo de translação é o valor mínimo, assim garante que o novo mínimo seja zero e o novo máximo seja um.

$$x'_i = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{16}$$

Para atributos sem outliers, o *Min-Max Scaler* produz um efeito semelhante ao *Standard Scaler*. Porém, quando há *outliers*, essa técnica não consegue equalizar médias e variâncias das distribuições, o que pode comprometer seu uso em pipelines de ML.

2.5.3 *Maximum Absolute Scaler*

Essa técnica modifica a escala de um atributo dividindo cada valor pelo valor absoluto máximo da variável, conforme a Equação 17.

$$x'_i = \frac{x_i}{\max(|x|)} \tag{17}$$

É sensível a *outliers*, pois a presença de valores extremos pode estreitar significativamente a distribuição dos dados normais.

2.5.4 *Normalizer Scaler*

É uma técnica que transforma um vetor de características de modo que sua norma seja igual a 1, geralmente usando a norma L2, como indicado na Equação 18 (HAMSICI; MARTINEZ, 2007).

$$\mathbf{x}' = \frac{\mathbf{x}}{\|\mathbf{x}\|} \tag{18}$$

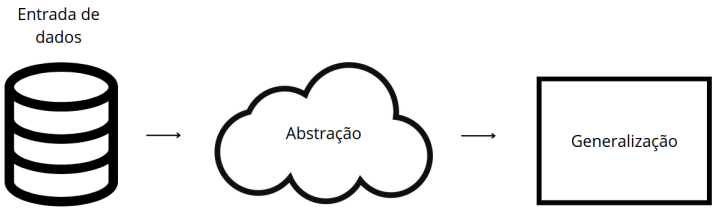
Dessa forma, cada vetor preserva a direção relativa entre seus elementos, descartando a magnitude absoluta. Uma limitação do *Normalizer* é que ele não lida diretamente com *outliers*, já que um valor extremo pode afetar a direção do vetor normalizado.

2.6 *MACHINE LEARNING* (ML)

Machine Learning (ML) é um campo da computação que estuda como as máquinas podem aprender e inferir informações a partir de dados, sem serem explicitamente programadas para tal tarefa (CHANDRAMOULI; DUTT; DAS, 2019). Desse modo, um programa aprenderia por uma experiência E a realizar uma tarefa T, de modo que seu desempenho D melhore com a experiência adquirida.

O processo de ML envolve três etapas principais, definidas na Figura 5.

Figura 5 – Etapas ML.



Fonte: produzida pela autora com influência de (CHANDRAMOULI; DUTT; DAS, 2019).

1. Entrada de dados: recebe dados relacionados à tarefa.
2. Abstração: criação de um modelo que compreende os padrões relevantes nos dados.
3. Generalização: aplicação do modelo para tomar decisões ou fazer previsões em novos dados desconhecidos.

Para o sucesso, a escolha do modelo matemático e probabilístico que mais se encaixa a tarefa é fundamental, sendo que os parâmetros de diferentes modelos podem ser modificados de modo a adaptá-los cada vez mais àquela circunstância do enunciado e das informações analisadas.

2.6.1 K-Vizinhos Mais Próximos (KNN)

O KNN é um algoritmo de aprendizado supervisionado usado em tarefas de classificação. No aprendizado supervisionado, o modelo é treinado com dados rotulados (dados de entrada e saída já conhecidos) (NASTESKI, 2017).

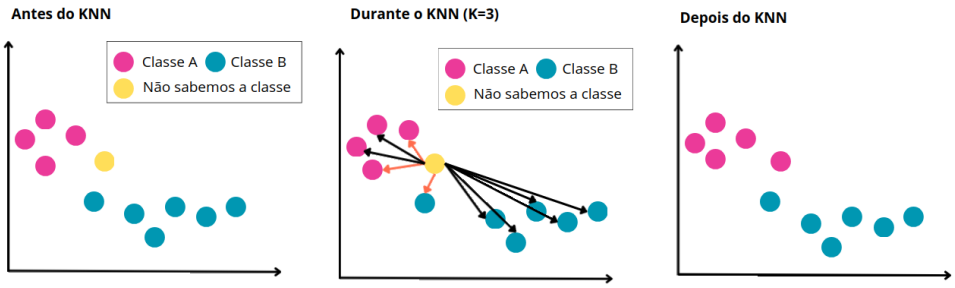
No caso do KNN é um pouco diferente, ele não exige um treinamento prévio com um conjunto específico de dados, sendo conhecido como um método de *lazy-learning*. Esse nome se dá porque ele não cria um modelo interno ao longo do treinamento, sem uma fase de treino explícita.

Durante sua execução, o algoritmo calcula a distância da amostra a ser classificada em relação a cada elemento do conjunto de "treinamento", ou seja, o conjunto com dados rotulados. Assim, ele identifica os K (pode ser modificado até achar o de melhor valor) elementos conhecidos mais próximos do dado desconhecido. Por fim, ele conta qual o rótulo mais comum no conjunto de K elementos, atribuindo o mesmo valor para a classe do dado desconhecido (SUYAL; GOYA,

2022). É simples e eficaz, sendo aplicado em diferentes problemas que envolvem reconhecimento de padrões (PR).

Na Figura 6, ilustra-se esse procedimento, mostrando um dado desconhecido (amarelo), que será a amostra de entrada. Depois, ele calcula a distância dessa amostra em relação a todos os outros dados no espaço (setas). Com $K = 3$, ele seleciona os três vizinhos mais próximos (setas laranjas), identificando a amostra como pertencente à classe A (rosa).

Figura 6 – Passo a passo do KNN.



Fonte: produzido pelos autores.

A validação cruzada (*K-Fold Cross Validation*) é uma técnica que pode ser aplicada para avaliar a generalização e robustez do KNN. Para isso, o conjunto de dados é dividido em n subconjuntos, de modo que, em cada iteração, um seja usado como teste e os demais como treino, repetindo-se o processo n vezes (GORRIZ et al., 2024). Essa abordagem permite analisar o modelo sob diferentes configurações e detectar possíveis casos de sobre-ajuste (*overfitting*), quando o desempenho em dados novos é inferior ao desempenho em partições específicas.

O valor k é um dos hiperparâmetros ajustáveis mais importantes do algoritmo, uma vez que um valor muito baixo pode tornar o modelo sensível a ruídos, enquanto um valor alto podem reduzir demais as fronteiras entre classes. Ademais, se o valor de k for par, deve haver um critério de desempate entre classes.

Outro hiperparâmetro relevante é a métrica utilizada para calcular a distância entre os pontos, a qual depende da natureza e do tipo de dados. Conforme destacado no artigo "*Comparison of Distance Measurement Methods on K-Nearest Neighbor Algorithm for Classification*"de (ROSA; PRIMARTHA; WIJAYA, 2019), algumas das métricas mais comuns incluem a distância Euclidiana, a distância de Manhattan e a similaridade do cosseno.

2.6.1.1 Distância Euclidiana

A distância Euclidiana é definida como a raiz quadrada da soma dos quadrados das diferenças entre as coordenadas correspondentes dos pontos $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ e $\mathbf{y} = (y_1, y_2, \ldots, y_n)$, como na Equação 19.

$$d_{\text{Euclidiana}}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{19}$$

Essa métrica mede a distância em linha reta entre os pontos, sendo mais adequada para dados numéricos contínuos, pois captura melhor suas variações. Além disso, é indicada quando as variáveis estão em escalas homogêneas, evitando que algum atributo tenha peso desproporcional no cálculo da distância.

2.6.1.2 Distância Manhattan

É a soma das diferenças absolutas entre as coordenadas dos pontos, como na Equação 20.

$$d_{\text{Manhattan}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i| \tag{20}$$

Assim como a distância Euclidiana, a Manhattan funciona melhor com dados numéricos contínuos. Além disso, por não elevar as diferenças ao quadrado, cada atributo contribui de forma proporcional para a distância, tornando o método menos sensível a *outliers*.

2.6.1.3 Similaridade do Cosseno

A similaridade do cosseno mede a orientação relativa de dois vetores, avaliando sua direção mais do que sua magnitude, sendo ela representada pela Equação 21.

$$\text{similaridade}_{\cos}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \tag{21}$$

Para usar como métrica de distância, pode-se definir $d_{\cos} = 1 - \text{sim}_{\cos}$. Essa métrica é útil quando a magnitude dos dados importa menos, sendo a orientação e direção a parte mais importante para compreendê-los no espaço.

2.7 COMPUTAÇÃO QUÂNTICA

A Computação Quântica é um ramo da computação que se baseia nos princípios da mecânica quântica, a qual descreve o comportamento da matéria e da energia em escalas subatômicas. Diferentemente da computação clássica, cuja unidade básica é o *bit* (0 ou 1), a Computação Quântica utiliza o *qubit*, capaz de representar múltiplos estados simultaneamente (NIELSEN; CHUANG, 2010). Essa característica permite executar cálculos mais rápido, alcançando o chamado "*quantum advantage*".

Apesar disso, ainda não se sabe até que ponto a Computação Quântica pode superar a clássica. Em algumas tarefas, como a fatoração de números grandes (impactando a criptografia RSA), já demonstram vantagem, mas ainda estamos na Era da "Computação Quântica Ruidosa de Escala Intermediária"(NISQ), com limitações de *hardware* que restringem a aplicação prática dessa tecnologia (PRESKILL, 2018).

Contudo, há diversas áreas em que a Computação Quântica já mostra alto potencial de pesquisa, incluindo química (CAO et al., 2018), biologia molecular (MINOCHA; NAMASUDRA, 2023), entre outros.

2.7.1 Qubits

As definições de *qubit* apresentadas nessa sub-seção são baseadas no livro "*Quantum Computation and Quantum Information*", de (NIELSEN; CHUANG, 2010).

Como já discutido, os *qubits* são a unidade fundamental da informação em um computador quântico. Diferentemente dos bits clássicos, que só podem assumir os valores 0 ou 1, os *qubits* podem existir em uma superposição de ambos os estados. Matematicamente, um *qubit* é representado por um vetor de estado na notação de Dirac, que utiliza os símbolos $|0\rangle$ e $|1\rangle$. Seu estado geral pode ser descrito como uma combinação linear de ambos os estados, conforme mostrado na Equação 22.

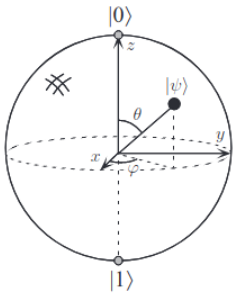
$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{22}$$

Os valores α e β são números complexos chamados "amplitudes de probabilidade", que satisfazem a condição de normalização da Equação 23.

$$|\alpha|^2 + |\beta|^2 = 1 \tag{23}$$

Essas amplitudes determinam as probabilidades de medir o *qubit* nos estados $|0\rangle$ ou $|1\rangle$. Embora um *qubit* possa conter inúmeras informações complexas, uma medição colapsa seu estado, revelando apenas um bit. O estado do *qubit* pode ser visualizado geometricamente na esfera de Bloch, onde os ângulos θ e φ definem sua posição, como na Figura 7.

Figura 7 – Representação do *qubit* na esfera de Bloch.



Fonte: (NIELSEN; CHUANG, 2010)

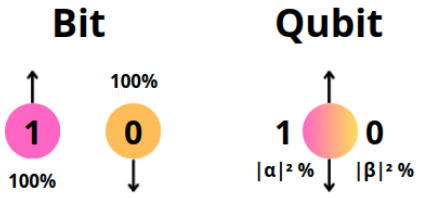
2.7.2 Superposição

Em termos gerais, superposição refere-se à combinação de dois ou mais estados de um sistema, permitindo que ele apresente características de todos simultaneamente (DIRAC, 1958). Embora também exista em sistemas clássicos, como em computadores ópticos (KHRENNIKOV,

2021), na mecânica quântica a superposição é probabilística, colapsando para um único estado apenas quando o sistema é medido.

Na Figura 8, mostra-se um exemplo, em que o sistema tem $|\alpha|^2\%$ de probabilidade de estar no estado $|0\rangle$ e $|\beta|^2\%$ de probabilidade de estar no estado $|1\rangle$.

Figura 8 – Representação da superposição de *qubits*.



Fonte: produzido pelos autores.

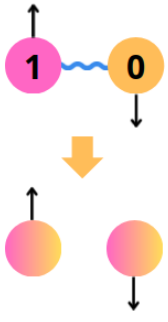
Essa propriedade permite que múltiplos estados sejam explorados em paralelo, com o número de combinações crescendo exponencialmente com a quantidade de *qubits* (2^n). Ao medir o sistema, o colapso probabilístico seleciona um estado, possibilitando que algoritmos quânticos encontrem soluções ótimas ou significativamente melhores do que métodos clássicos em certos problemas.

2.7.3 Emaranhamento Quântico

O emaranhamento quântico (*entanglement*) ocorre quando um par ou grupo de *qubits* compartilha um estado quântico conjunto, de modo que a medição de metade dos *qubits* determina instantaneamente o estado dos demais (SANATI; BORZOEI, 2024). Ou seja, mudanças em um *qubit* afetam imediatamente os emaranhados a ele, permitindo um processamento paralelo.

Na Figura 9, há um exemplo de emaranhamento entre dois *qubits*. Cada *qubit* está em superposição de $|\alpha|^2\%$ de $|0\rangle$ e $|\beta|^2\%$ de $|1\rangle$, mas devido ao emaranhamento, seus estados são correlacionados. Assim, medir um deles determina instantaneamente o estado do outro, independentemente da distância, acelerando o processamento de informações.

Figura 9 – Representação do emaranhamento de *qubits*.



Fonte: produzido pelos autores.

2.7.4 Operações

Os *qubits* são manipulados por portões quânticos, que correspondem a matrizes unitárias complexas capazes de alterar o estado da partícula em nível subatômico.

Entre eles, destacam-se os portões de um único *qubit* (*single-qubit gates*), como Identidade (I), Pauli (X, Y, Z), Hadamard (H), de fase (S, T) e de rotação (R_x, R_y, R_z) (CHOWDHURY; NATH, 2022).

Portões de múltiplos *qubits* (*multi-qubit gates*) atuam simultaneamente em dois ou mais *qubits*, combinando seus estados via produto tensorial ($\otimes$), permitindo operações mais complexas (CHOWDHURY; NATH, 2022).

2.7.4.1 Portão de rotação em Y (R_y)

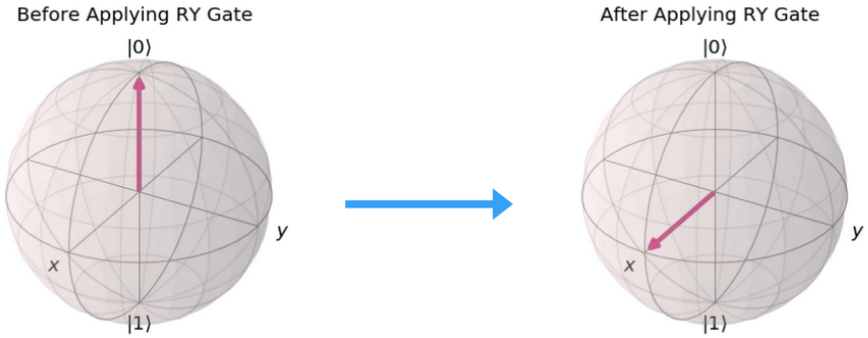
Neste trabalho, a única operação quântica utilizada é a rotação em torno do eixo y, implementada pelo portão R_y (CHOWDHURY; NATH, 2022). Esse portão altera a posição de um *qubit* na esfera de Bloch, controlando a proporção de superposição entre os estados $|0\rangle$ e $|1\rangle$ por meio de rotações ao longo do eixo y.

A matriz unitária correspondente é apresentada na Equação 24. Na matriz, θ é o ângulo de rotação.

$$R_y(\theta) = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix} \tag{24}$$

Na Figura 10, demonstra-se como uma rotação em y com $\theta = \frac{\pi}{2}$ funciona.

Figura 10 – Representação da operação $R_y(\theta)$.



Fonte: (DEEP LEARNING UNIVERSITY, 2025).

2.7.5 Medidas

Em sistemas quânticos, a medição de um *qubit* colapsa seu estado quântico em um valor clássico da base computacional, como $|0\rangle$ ou $|1\rangle$. Existem diferentes formas de realizar medições nesses sistemas (NETO et al., 2025), incluindo valor esperado, amostras, contagens e probabilidades.

Nesta pesquisa, foi usada apenas a medição por valor esperado para calcular a similaridade entre estados quânticos no kernel aplicado no QKNN. O valor esperado representa a média de um observável quântico em um dado estado, funcionando como uma medida estatística. Em outras palavras, o valor esperado expressa a média ponderada dos possíveis resultados de medição, considerando as probabilidades associadas a cada um, fornecendo uma estimativa estatística do comportamento médio do sistema.

2.7.6 Circuitos quânticos

Os circuitos quânticos são uma forma de descrever algoritmos quânticos, consistindo em operadores (portões) aplicados a *qubits*, que podem ser manipulados individualmente ou em conjunto. Cada circuito tem um custo computacional, em termos de número total de portões e profundidade (NIELSEN; CHUANG, 2010).

Os circuitos permitem visualizar e planejar operações em múltiplos *qubits*, incluindo operações controladas e estados emaranhados, sendo essenciais para desenvolver algoritmos quânticos com os fins específicos da aplicação desejada.

2.7.7 Classificadores quânticos

Quantum Machine Learning (QML) é um campo que combina Computação Quântica e Inteligência Artificial (IA), buscando aproveitar fenômenos quânticos como superposição e emaranhamento para melhorar o desempenho de algoritmos de ML (SANATI; BORZOEI, 2024).

É um campo de pesquisa recente que vem avançando bastante nos últimos anos. Atualmente, já existem aplicações de Redes Neurais Quânticas (QNNs) e versões quânticas de diferentes tipos de algoritmos de ML (QML). Um grande desafio dessa área é a presença de ruídos, que interferem na qualidade das classificações e impedem o alcance do "*quantum advantage*"na Era NISQ (PRESKILL, 2018).

2.8 MÉTODOS ESTATÍSTICOS

As métricas estatísticas numéricas de avaliação de desempenho utilizadas nesta pesquisa são baseadas no artigo "*A systematic analysis of performance measures for classification tasks*", de (SOKOLOVA; LAPALME, 2009), que estudou a aplicação de avaliações estatísticas em problemas multi-classes.

Para a explicação das métricas, adota-se adotar como padrão: TP são os verdadeiros positivos, TN os verdadeiros negativos, FP os falsos positivos e FN os falsos negativos.

2.8.1 Acurácia

Representa o total de acertos dentre todas as classes, como representado na Equação 25.

$$Acurcia = \frac{TP + TN}{TP + FP + TN + FN} \tag{25}$$

Contudo, a acurácia é uma métrica tendenciosa em cenários de classes desbalanceadas, pois pode apresentar valores altos mesmo quando o modelo falha sistematicamente em prever classes minoritárias. No caso deste trabalho, todas as classes são balanceadas, porém, outras métricas foram aplicadas para melhorar a análise final do desempenho.

2.8.2 Precisão

Mede a proporção de exemplos que foram classificados como pertencentes a uma determinada classe que realmente pertencem a essa classe. Ou seja, é a confiabilidade do modelo ao predizer uma classe específica. Quanto maior a precisão, menor o número de falsos positivos, como na Equação 26.

$$\text{Precisão} = \frac{TP}{TP + FP} \tag{26}$$

2.8.3 *Recall*

Também conhecido como sensibilidade, o *recall* mede a capacidade do modelo de identificar corretamente todos os exemplos que pertencem a uma determinada classe. Ou seja, dos exemplos que realmente são daquela classe, quantos o modelo conseguiu determinar de forma correta, como apresentado na Equação 27.

É aplicado quando o objetivo é minimizar falsos negativos.

$$Recall = \frac{TP}{TP + FN} \tag{27}$$

2.8.4 *F1-score*

O *F1-score* é uma métrica que combina a precisão e *recall* em uma única medida. Ele é útil para avaliar o desempenho do modelo quando há um equilíbrio entre precisão e *recall*, como em classificações com dados desbalanceados.

A fórmula do *F1-score* para uma classe é dada pela Equação 28:

$$F_1 = 2 \cdot \frac{\text{Precisão} \times \text{Revocação}}{\text{Precisão} + \text{Revocação}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{28}$$

2.8.5 Taxa de erro

A taxa de erro é a métrica complementar da acurácia e representa a proporção de classificações incorretas feitas pelo modelo, em relação ao total de amostras avaliadas, como na Equação 29.

$$\text{Taxa de Erro} = 1 - \text{Acurácia} = \frac{FP + FN}{TP + FP + TN + FN} \tag{29}$$

2.8.6 *Box-plot*

É um gráfico que apresenta a distribuição de um conjunto de dados de forma resumida, mostrando a mediana, os quartis (Q1 e Q3), os valores mínimo e máximo, e possíveis anomalias. É útil para visualizar a dispersão dos dados e identificar valores extremos.

2.8.7 *Macro Average*

Em problemas de classificação multi-classe, métricas de avaliação como precisão, *recall* e *F1-score* podem ser calculadas de diferentes maneiras para resumir o desempenho do modelo considerando todas as classes. Um dos meios mais comum é o *Macro Avarage*, que utiliza o valor das métricas individuais por classe e faz uma média ponderada com mesmo peso para cada, assim o desempenho geral do algoritmo pode ser obtido. Quando há desbalanceamento entre classes, o peso igual pode prejudicar a análise, porém isso não acontece neste trabalho, já que os dados são balanceados, logo essa técnica será aplicada nas análises.

2.8.8 Matriz de confusão

É uma ferramenta que auxilia na observação do desempenho do algoritmo por classe, apresentando uma tabela que relaciona as classes reais (linhas) com as classes previstas pelo modelo (colunas), permitindo visualizar os acertos e os erros de classificação. Os valores na diagonal principal são os acertos, enquanto os outros demonstram confusões entre classes que o modelo cometeu.

3 TRABALHOS CORRELACIONADOS

Neste capítulo, são apresentados trabalhos relacionados ao tema que fundamentam esta pesquisa científica, abrangendo tanto abordagens clássicas quanto quânticas.

3.1 USO DO KNN NA CLASSIFICAÇÃO DE VOCALIZAÇÕES DE AVES

Na literatura, diversos estudos exploram a bioacústica e o processamento digital de sinais (PDS) para a identificação de espécies de aves. O algoritmo KNN, devido à sua simplicidade e eficiência, tem sido aplicado e analisado em pesquisas que investigam como diferentes métodos de extração e tratamento de dados influenciam o desempenho na classificação das espécies.

Nesse contexto, Bang e Rege (2013), no artigo "*Classification of Bird Species based on Bioacoustics*", exploraram a extração de características a partir da energia do sinal em diferentes bandas de frequência, utilizando o KNN para classificar as espécies e avaliar o desempenho do método. Os autores relataram bons resultados, alcançando 91% de acurácia para oito espécies, demonstrando que essas características energéticas podem ser discriminantes mesmo com modelos simples (BANG; REGE, 2013).

Em 2017, os mesmos autores, no estudo "*Recognition of Bird Species from their Sounds using Data Reduction Techniques*", aplicaram o KNN em conjunto com coeficientes cepstrais na escala Mel (MFCCs) e técnicas de redução de dimensionalidade, como PCA e quantização vetorial. O trabalho alcançou 82% de acurácia na classificação de dez espécies, demonstrando que, com extração e redução adequadas, o KNN pode obter bom desempenho (BANG; REGE, 2017). No estudo de 2017, eles aumentaram o número de classes em relação ao anterior, notando que a quantidade de classes reduz a capacidade do algoritmo, algo a ser trabalhado em pesquisas semelhantes.

Em 2023, Wu, Kosuru e Tippareddy, no artigo "*Bird Species Identification from Audio Data*", aplicaram diferentes técnicas de pré-processamento e extração de características, incluindo MFCC, ZCR e outras medidas espectrais, com o objetivo de classificar vocalizações de aves. Foram avaliados vários algoritmos de ML supervisionados, como KNN, SVC, SGD, *Decision Tree*, *Random Forest* e *Naïve Bayes* (WU; KOSURU; TIPPAREDDY, 2023). Entre eles, o KNN apresentou desempenho competitivo, alcançando 71% de acurácia na classificação de três espécies, demonstrando eficácia em cenários com número reduzido de classes.

No entanto, os autores observaram que a acurácia diminui à medida que o número de espécies aumenta, devido à semelhança espectral entre os cantos e à variabilidade acústica intra-espécie, limitando a escalabilidade do modelo. Por exemplo, em um cenário com 30 classes, o KNN atingiu apenas 28% de acurácia, ainda assim próximo do desempenho dos demais modelos testados. O que indica que o uso do KNN para problemas multi-classes e para classificação de aves com mais de um tipo de canto é um desafio.

Por fim, Pattraksha et al. (2023) aplicaram o KNN para a classificação multi-classes de espécies de pássaros a partir de gravações de suas vocalizações. Para extração de características, utilizaram MFCC, devido ao seu sucesso em trabalhos anteriores para a mesma tarefa. O desempenho do modelo foi avaliado com e sem validação cruzada: sem validação cruzada, obteve-se 80% de acurácia, enquanto com validação cruzada, a acurácia foi de 77% para o melhor valor de k, $k = 1$ (PATTRAKSHA et al., 2023).

3.2 USO DO OPERADOR DE *TEAGER–KAISER* EM BIOACÚSTICA

O estudo que utilizou o Operador de *Teager–Kaiser* em bioacústica, servindo de inspiração para sua aplicação nesta pesquisa, foi *"Automatic detection of echolocation clicks based on a Gabor model of their waveform"*, de Madhusudhana et al. (2015). Nele, o operador de *Teager–Kaiser* foi um dos algoritmos aplicados durante um pipeline voltado para a detecção de sons de ecolocalização de mamíferos marinhos (MADHUSUDHANA; GAVRILOV; ERBE, 2015). No trabalho, o método obteve bom desempenho, destacando-se por sua simplicidade e leveza computacional, sendo que o uso do TKEO aprimorou as inferências dos resultados.

3.3 *MACHINE LEARNING* QUÂNTICO (QML)

O uso de algoritmos de ML quânticos tem se tornado cada vez mais frequente. Assim, busca-se avaliar se esse paradigma oferece vantagens em tarefas de classificação, constituindo um campo de pesquisa recente e com questões ainda em aberto.

Em 2013, Aïmeur, Brassard e Gambs foram um dos pioneiros ao abordar o tema no artigo "*Quantum speed-up for unsupervised learning*", no qual propuseram algoritmos de clusterização com sub-rotinas quânticas (árvore geradora mínima, *K-medians* e agrupamento divisivo) (AIMEUR; BRASSARD; GAMBS, 2013). Nesse sentido, concluíram que, ao integrar partes quânticas, era possível acelerar tarefas de aprendizado não supervisionado, mas ainda enfrentaram com desafios associados à redução de dimensionalidade dos modelos.

Assim, com o avanço da Computação Quântica, surgiram mais variações nesse paradigma de algoritmos clássicos de ML, os chamados algoritmos híbridos. Em 2022, De Luca, no artigo "*A Survey of NISQ Era Hybrid Quantum-Classical Machine Learning Research*" (LUCA, 2022), apresentou uma revisão sobre o uso de ML híbrido (*Hybrid Quantum-Classical Machine Learning* – HQC) no contexto da Era NISQ (*Noisy Intermediate-Scale Quantum*). Essa Era, nomeada por Preskill (2018), caracteriza o estado atual da Computação Quântica, em que dispositivos possuem de dezenas a algumas centenas de *qubits*, mas ainda carecem de correção de erros, principalmente os associados a ruídos ocasionados pelo *hardware*, gerando a perda das capacidades quânticas dos *qubits* por decoerência (PRESKILL, 2018). Nesse cenário, De Luca destaca o paradigma híbrido como uma das abordagens mais viáveis que vem sendo utilizada em aplicações quânticas na atualidade.

3.3.1 QML em aplicações ambientais

Com a possibilidade do uso de algoritmos híbridos, surgiram algumas aplicações ambientais que usam QML, investigando se suas capacidades de processar uma grande quantidade de informação em paralelo e inferir sobre dados complexos trazem vantagens reais nesse caso de uso.

Dessa forma, um estudo realizado por Santos et al. (2025) aplicou uma QML híbrida para previsão da concentração de ozônio no ar com até 6 horas de antecedência, atingindo resultados competitivos, que até superou modelos de previsão já consolidados (SANTOS et al., 2025).

Em outra aplicação, Rezaei e Javadi utilizaram QML (QSVM) comparado com ML (SVM) para analisar impactos ambientais causados por tecnologias oceânicas de geração de energia, como energia gerada por ondas e marés. No artigo, demonstraram que a QML superou o algoritmo de ML clássico em termos de acurácia, indicando que o uso de algoritmos quânticos é uma abordagem promissora para a análise de dados ambientais (REZAEI; JAVADI, 2024).

3.3.2 QKNN

O algoritmo de KNN em sua versão quântica já foi aplicado por meio de diferentes abordagens, tendo sido exploradas diversas formas de calcular distâncias. Assim, Zardini, Blanzieri e Pastorello (2024) foram um dos primeiros a propor um algoritmo de QKNN baseado na distância Euclidiana (ZARDINI; BLANZIERI; PASTORELLO, 2024), como será testado neste trabalho. Embora a distância Euclidiana seja muito comum em KNN clássico, ela é pouco explorada em sua versão quântica, como citado pelos autores. No artigo, eles propõem novas formas de codificação dos dados em amplitudes quânticas, criando um modelo simples, que exige poucos *qubits*. Os resultados mostraram que, em execução ideal, o algoritmo atinge desempenho equivalente ao clássico.

Logo, o uso da distância Euclidiana no QKNN mostra-se promissor e precisa de mais investigações. Por esse motivo, foi aplicado nesta pesquisa, que buscou desenvolver formas simples de implementar um QKNN baseado em distância Euclidiana, testando-o na identificação de cantos de aves a partir de dados bioacústicos, uma aplicação inédita para testes de QKNNs híbridos.

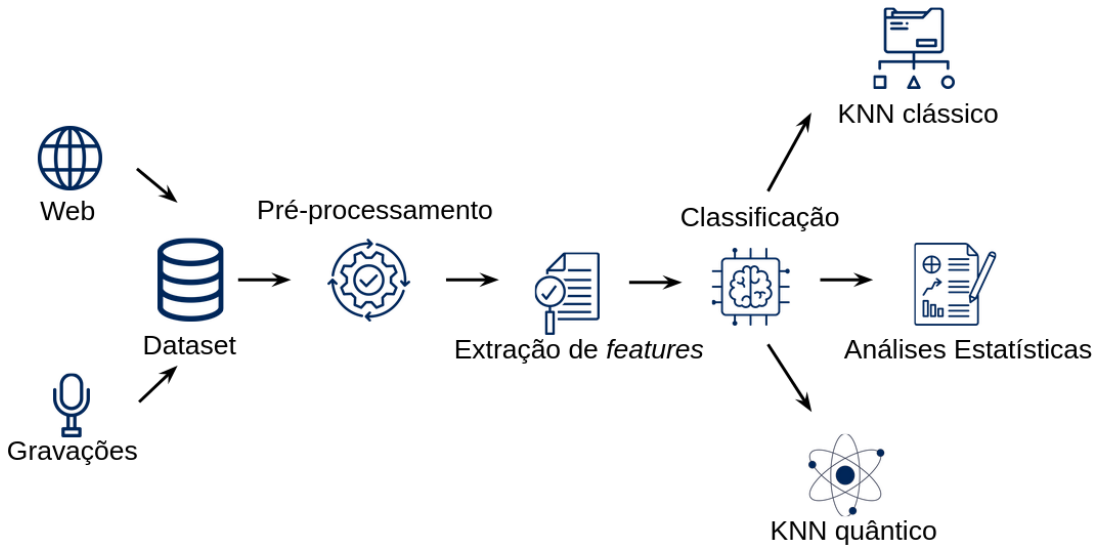
4 MATERIAIS E MÉTODOS

A metodologia adotada nesta pesquisa consistiu em um conjunto estruturado de etapas que conduziram ao resultado final, como descrito a seguir:

1. Coleta e tratamento dos dados;
2. Pré-processamento;
3. Extração de *features*;
4. Aplicação do KNN clássico;
5. Aplicação do KNN quântico;
6. Análise estatística e comparativa dos resultados.

Na Figura 11, apresenta-se um fluxograma que sintetiza as etapas acima. Ao longo deste capítulo, cada uma delas será detalhada individualmente.

Figura 11 – Fluxograma da metodologia implementada.



Fonte: Produzido pelos autores.

4.1 COLETA DE DADOS

O conjunto de dados utilizado nesta pesquisa foi composto por gravações de áudio referentes a 20 espécies de aves brasileiras, totalizando 1400 arquivos para o conjunto de dados rotulados (treinamento), e aproximadamente 600 arquivos para o conjunto de teste, no qual as espécies deveriam ser identificadas. Cada espécie foi representada por 70% gravações no conjunto de treinamento e 30% no de teste para as partições manuais.

As gravações foram obtidas de três formas: (i) por meio da plataforma Xeno-Canto, que disponibiliza um grande acervo de áudios de espécies de aves online e gratuitamente (XENO-CANTO, 2025), (ii) por gravação direta de aves da cidade e (iii) gravação de aves encontradas em outras fontes na internet, como *Youtube*. Os áudios obtidos por gravação direta foram produzidos com maior controle ambiental, apresentando menor incidência de interrupções ou ruídos externos, como vocalizações de espécies de fundo. Por outro lado, os áudios obtidos via *download* demandaram etapas adicionais de edição e corte, visando remover ruídos e padronizar a duração. Esses *downloads* foram realizados de forma automatizada por meio de um *script* em *Bash*.

As espécies que foram classificadas pelo modelo multi-classe de KNN são apresentadas na Figura 12.

Figura 12 – Espécies de aves brasileiras classificadas.

Fonte: Produzido pelos autores.

Vale ressaltar que 28% das amostras foram geradas por meio de *data augmentation*, feita em Linguagem *Python* por meio da biblioteca *librosa* (MCFEE et al., 2015). As operações aplicadas para ampliar os dados foram: adição de ruído gaussiano, alteração da velocidade (*time-stretching*) e mudança do tom (*pitch-shifting*).

4.1.1 Abertura dos arquivos

Todos os áudios estavam no padrão WAV, um subconjunto da especificação RIFF da *Microsoft* para multimídia (MICROSOFT, 2025). Esse padrão define um bloco com dois sub-blocos, um de cabeçalho com metadados do arquivo e um de dados com as informações vocais da ave.

Como foi usado C++, todas as funções de abertura foram construídas manualmente (sem *framework* específico), realizando as seguintes etapas:

- Identificação do formato: leitura do cabeçalho RIFF, verificando se o arquivo é realmente do tipo *RIFF/WAVE*.
- Leitura do sub-bloco *fmt*: indicava os parâmetros do áudio, como o tipo de codificação (PCM), número de canais (mono ou estéreo), taxa de amostragem em Hz, alinhamento de blocos e a taxa média de bytes por segundo.
- Tratamento de exceções: alguns arquivos continham bytes extras no bloco de formato, que o programa descartava para alinhar corretamente a leitura.
- Leitura do sub-bloco *data*: continha o tamanho e sequência de amostras do áudio, garantindo a amostragem correta.
- Extração das amostras: o programa lia as amostras em diferentes configurações (8 ou 16 bits, mono ou estéreo), convertia em valores numéricos *double* e armazenava em memória para começar o processamento.

Após a leitura dos dados, as amostras foram enviadas para o pré-processamento, a primeira etapa de polimento e tratamento dos dados.

4.2 PRÉ-PROCESSAMENTO

Além da remoção de trechos irrelevantes, do corte de segmentos com múltiplas aves cantando simultaneamente e da divisão dos áudios em partes menores (procedimentos realizados manualmente antes da abertura dos arquivos), foram aplicadas etapas adicionais para melhorar a manipulação das gravações pelo algoritmo.

4.2.1 Remoção da Componente DC

Em cada vetor de dados, removeu-se a componente DC, que desloca o sinal em relação ao eixo central (zero em y). A supressão do deslocamento no eixo vertical é importante para evitar distorções no espectro, garantindo que variações relevantes sejam analisadas consistentemente para todos os áudios, oscilando igualmente entre valores positivos e negativos.

Para isso, a função apresentada na Listagem 4.1 recebe o sinal de áudio por referência (vetor), garantindo a modificação direta de seus elementos. Primeiro, calcula-se a média dos valores do

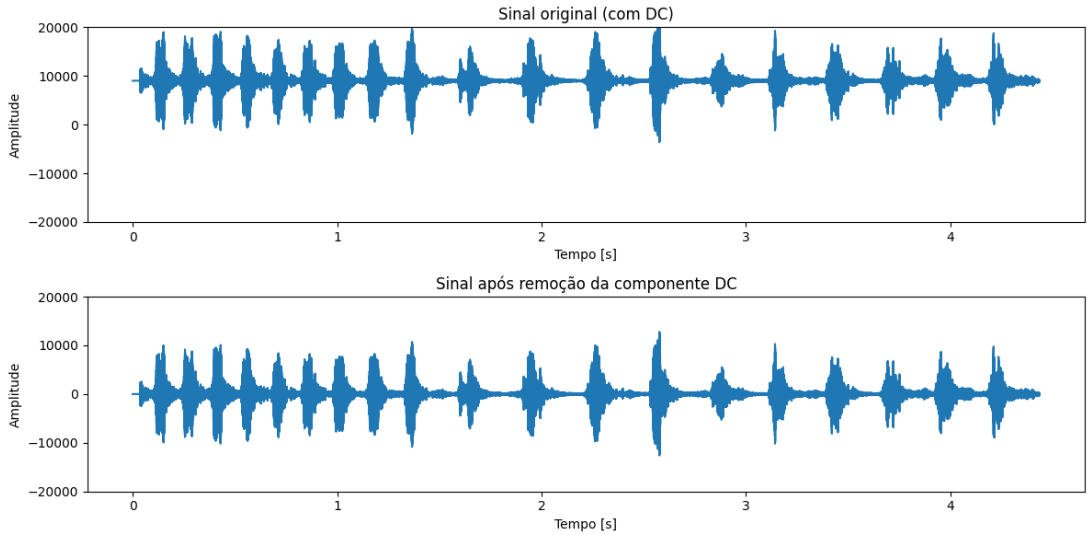
vetor, posteriormente subtraindo-na de cada componente do áudio. Isso assegura que o resultado final fique com média zero, o que deixa as amostras centralizadas.

```cpp
void remove_DC (vector<double> &audio)
{
    double media = 0.0;
    for (size_t i = 0; i < audio.size(); i++)
    {
        media += audio[i];
    }
    media /= audio.size();
    for (size_t i = 0; i < audio.size(); i++)
    {
        audio[i] -= media;
    }
}
```

Listing 4.1 – Função de remoção da componente DC em C++

Na Figura 13, é possível observar a aplicação da função de remoção do componente DC em um áudio de joão-de-barro. Nota-se que o sinal estava, originalmente, deslocado com seu centro próximo de 10000, mas após aplicar a função, ele fica centralizado em 0. Os gráficos foram plotados a partir dos valores gerados em C++, mas utilizando linguagem *Python*.

Figura 13 – Áudio de joão-de-barro antes e depois da remoção da componente DC.



Fonte: Produzido pelos autores.

4.2.2 Normalização do áudio

Como discutido, a normalização tem como objetivo o ajuste da amplitude do áudio, comprimindo seus valores para que fiquem dentro de uma faixa pré-determinada, geralmente entre 0 e

1, ou -1 e 1. Isso garante uma escala uniforme, independentemente da intensidade original da gravação, evitando que valores muito altos ou muito baixos influenciem desproporcionalmente na análise, o que mantém os resultados mais consistentes.

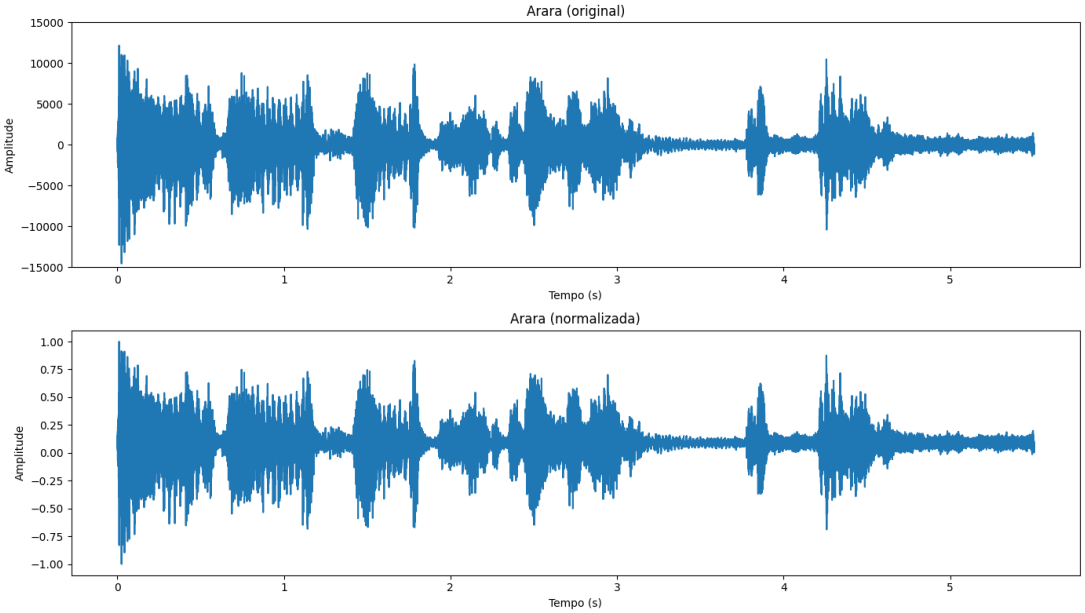
A normalização do áudio também foi implementada em C++, sendo aplicada antes e depois da extração de características. A normalização pré-extração tinha como objetivo padronizar a amplitude dos sinais, garantindo que valores de componentes obtidos fossem consistentes entre as amostras. Já a normalização pós-extração era empregada sobre as características, assegurando que cada dimensão tivesse magnitude comparável, o que é fundamental para algoritmos sensíveis à escala, como o KNN. No pré-processamento, adotou-se como padrão a normalização mínimo-máximo na faixa -1 e 1, que mostrou bom desempenho antes da aplicação da extração de características.

A função em C++ recebe o vetor de áudio por referência e aplica a normalização do tipo mínimo-máximo conforme a Equação 16. Embora essa fórmula produza valores no intervalo $[0, 1]$, o código realiza um reescalonamento adicional para ajustar os resultados ao intervalo $[-1, 1]$, utilizando a Equação 1.

$$x_i'' = 2x_i' - 1 \tag{1}$$

Na Figura 14, observa-se a amplitude do áudio antes da normalização, que ia de -15000 até $+15000$. Após a aplicação da normalização, o sinal fica achatado entre -1 e 1, mantendo suas características originais.

Figura 14 – Áudio da arara antes e depois da normalização mínimo-máximo.



Fonte: Produzido pelos autores.

4.3 EXTRAÇÃO DE FEATURES

Depois do pré-processamento dos áudios, foram extraídas apenas as características relevantes para a classificação. Diferentes métodos e combinações foram aplicados com o objetivo de identificar a configuração com melhor desempenho.

4.3.1 ZCR

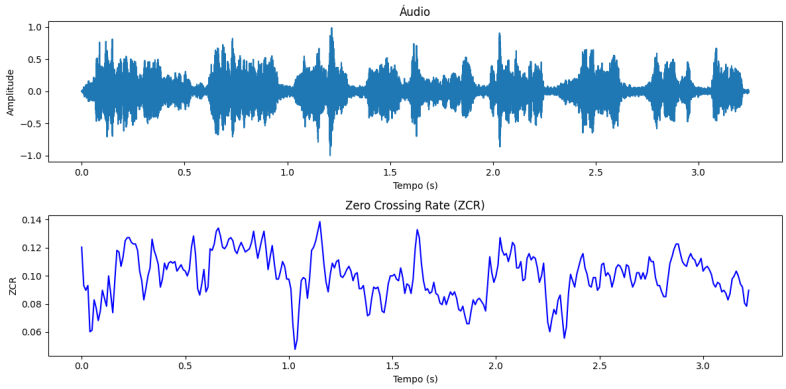
O primeiro método implementado foi o ZCR, desenvolvido manualmente em C++. A partir dele, era possível verificar quantas vezes a amostra trocava de sinal, auxiliando na identificação de sons suaves ou trocas abruptas durante as vocalizações.

A função ZCR implementada neste trabalho recebia o vetor com os dados originais do áudio e retornava um vetor com valores de ZCR calculados por *frames*. O cálculo era feito dividindo a amostra em blocos de tamanho fixo, os *frames*, com um certo valor de deslocamento entre eles. Para cada *frame*, o algoritmo contava o número de cruzamentos no eixo zero, ou seja, quantas vezes houve troca de sinal. O tamanho de cada *frame* era calculado a partir da duração desejada em milissegundos ($25ms$) e da taxa de amostragem do áudio. O resultado foi então normalizado pelo número de intervalos do *frame* $M - 1$, seguindo a definição matemática de ZCR dada pela Equação 3, no Capítulo 2. O procedimento foi implementado dessa forma para permitir que cada *frame* tivesse uma descrição mais detalhada do sinal, o que trouxe maior representatividade para o método.

Os vetores resultantes do cálculo do ZCR por *frames* apresentavam tamanhos diferentes (entre 800 e 1500 componentes) devido à variação de duração dos áudios. Para possibilitar o cálculo das distâncias no KNN, aplicou-se interpolação linear para igualar os vetores, garantindo comparabilidade entre amostras. Esse método foi escolhido pelo bom desempenho observado nos testes, embora outras padronizações de tamanho também fossem possíveis.

Na Figura 15, mostra-se um áudio de arara antes e depois da extração de ZCR. Os valores no gráfico representam a frequência de cruzamentos pelo eixo zero em cada frame, destacando trechos com maior ou menor atividade sonora, permitindo analisar a dinâmica do canto.

Figura 15 – Áudio de arara antes e depois da aplicação de ZCR.



Fonte: Produzido pelos autores.

4.3.2 Método de extração por energia acumulada

O método foi implementado em C++ e envolvia duas funções principais: (i) a primeira função calculava a energia total do sinal sonoro, percorrendo todos os valores do vetor original e aplicando a Equação 7 apresentada no Capítulo 2; (ii) a segunda utilizava essa energia total para gerar partições energéticas representativas do sinal.

A função apresentada na Listagem 4.2 definia inicialmente o parâmetro C, que determina em quantas partes o vetor será dividido para o cálculo da energia acumulada (granularidade da análise). Em seguida, calcula-se a energia total do áudio e estabelece-se a quantidade de energia que cada fração $\frac{C}{100}$ deve representar. O algoritmo percorre o vetor, acumulando energia até atingir o valor alvo de cada segmento, e armazena no vetor de *feature* a fração do sinal necessária para atingir a energia acumulada. Dessa forma, é gerado um perfil energético da gravação, usado para identificação de cada canto.

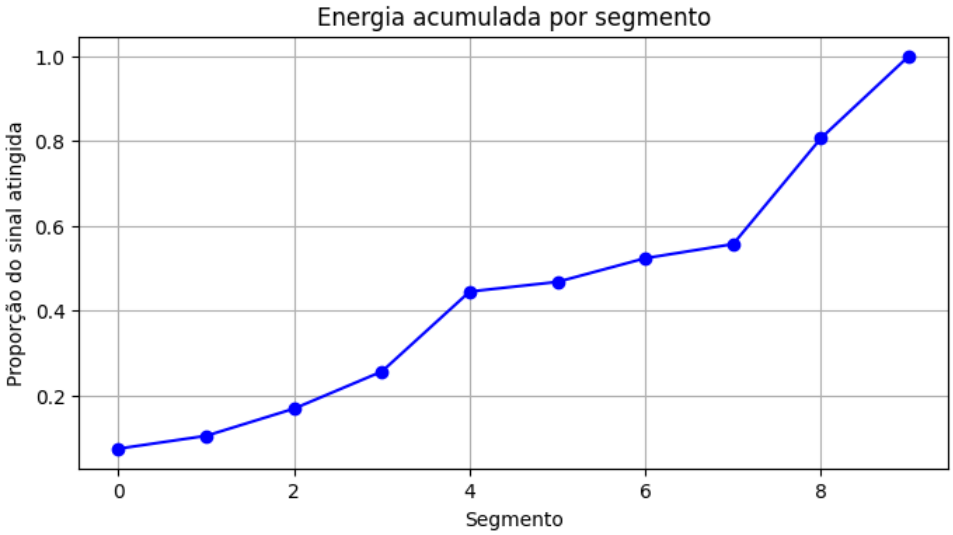
Na Listagem 4.2, `tam_parciais` indica a posição atual no vetor de áudio durante a acumulação de energia, enquanto `quantidade_vetores` define quantos valores serão armazenados no vetor de *features*. Na expressão lógica em `quantidade_vetores`, se o C for um inteiro, subtrai 1 da divisão para evitar um elemento a mais no vetor; caso não, já arredonda para baixo automaticamente com o *casting*.

```
1  vector<double> extracao_features_energia(vector<double> audio)
2  {
3      size_t tam_parciais = 0;
4      double C = 10.0;
5      int quantidade_vetores = ((100 / C) - ((int)(100 / C)) == 0) ?
           (100 / C) - 1 : (int)(100 / C);
6      vector<double> feature(quantidade_vetores);
7      double z = calculo_energia(audio) * ((double)((C) / 100));
8      double energia_acumulada = 0.0;
9      for (size_t i = 0; i < quantidade_vetores; i++) {
10         while (energia_acumulada < ((i + 1) * z))
11         {
12             energia_acumulada += audio[tam_parciais] * audio[
                   tam_parciais];
13             tam_parciais++;
14         }
15         feature[i] = static_cast<double>(tam_parciais) / static_cast<
               double>(audio.size());
16     }
17     return feature;
18 }
```

Listing 4.2 – Função de extração de características por energia acumulada em C++

Na Figura 16, mostra-se graficamente o resultado da aplicação da extração de características por energia acumulada em um áudio do canto da Saíra-sete-cores. No eixo x, estão representados os segmentos do áudio (de 1 a 10, definidos pelo valor de C), enquanto no eixo y indica-se a proporção de energia acumulada até cada segmento (de 0 a 1, com 1 sendo 100%). Observa-se que, até o segmento 4, já se acumula cerca de 42% da energia total, mostrando um crescimento parabólico no início. Entre 4 e 7, o crescimento da energia é menor. Por fim, entre os segmentos 7 e 10, ocorre um incremento expressivo, indicando mais intensidade no canto.

Figura 16 – Imagem representativa da energia acumulada ao longo do tempo no áudio da saíra.



Fonte: Produzido pelos autores.

4.3.3 Operador de *Teager*

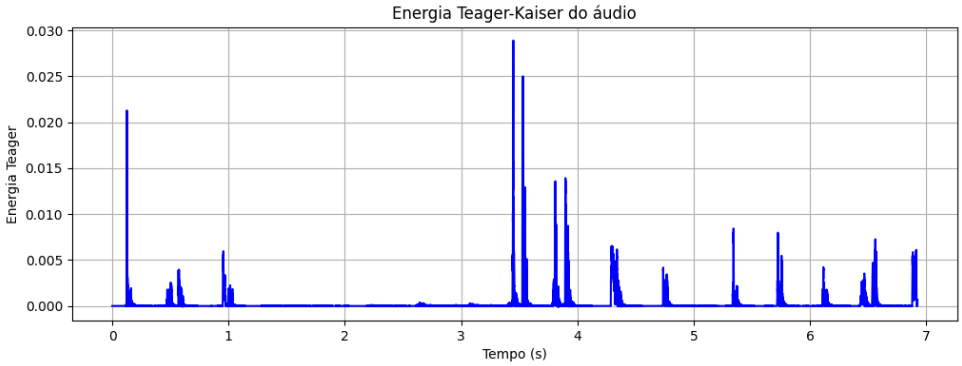
Para aplicar o operador de *Teager*, uma função em C++ recebia o vetor de dados por cópia e percorria cada elemento, calculando o operador em relação aos seus elementos vizinhos, conforme definido no Capítulo 2, na Equação 6, permitindo identificar variações rápidas de energia ao longo da amostra.

O método foi implementado tanto isoladamente quanto em conjunto com os demais; entretanto, quando utilizado sozinho, apresentou desempenho muito inferior, como será detalhado no Capítulo 5, em Resultados.

O tamanho dos vetores finais também é variável ao aplicar o *Teager*. Por isso, utilizou-se a interpolação para padronizar o comprimento das características.

Na Figura 17, tem-se o gráfico do resultado da aplicação da função que calcula o *Teager* em uma amostra de Canário-da-terra-verdadeiro. O eixo x representa o tempo, ou seja, o domínio sobre o qual a *feature* foi calculada. O eixo das ordenadas indica a energia de *Teager* do sinal ao decorrer de x, que reflete a intensidade instantânea do áudio. Os picos de energia indicam momentos em que o canto é intenso, ou seja, trechos de maior atividade e variações rápidas do sinal.

Figura 17 – Representação gráfica da Energia de *Teager* aplicada a um áudio de canário-da-terra-verdadeiro.



Fonte: Produzido pelos autores.

4.3.4 *Mel-Frequency Cepstral Coefficients* (MFCC)

O MFCC foi construído manualmente, seguindo o procedimento descrito na Seção 2.4.4 do Capítulo 2. Cada etapa foi implementada em C++, com funções específicas para cada operação. A implementação manual permitiu maior controle sobre os parâmetros, como tamanho das janelas, sobreposição, tipo de janela e número de filtros Mel, garantindo consistência na extração das características entre diferentes sinais.

Os parâmetros adotados (N = 256 amostras por quadro, M = 100 de sobreposição e extração dos 13 primeiros coeficientes) foram escolhidos por apresentarem bom desempenho na diferenciação entre espécies.

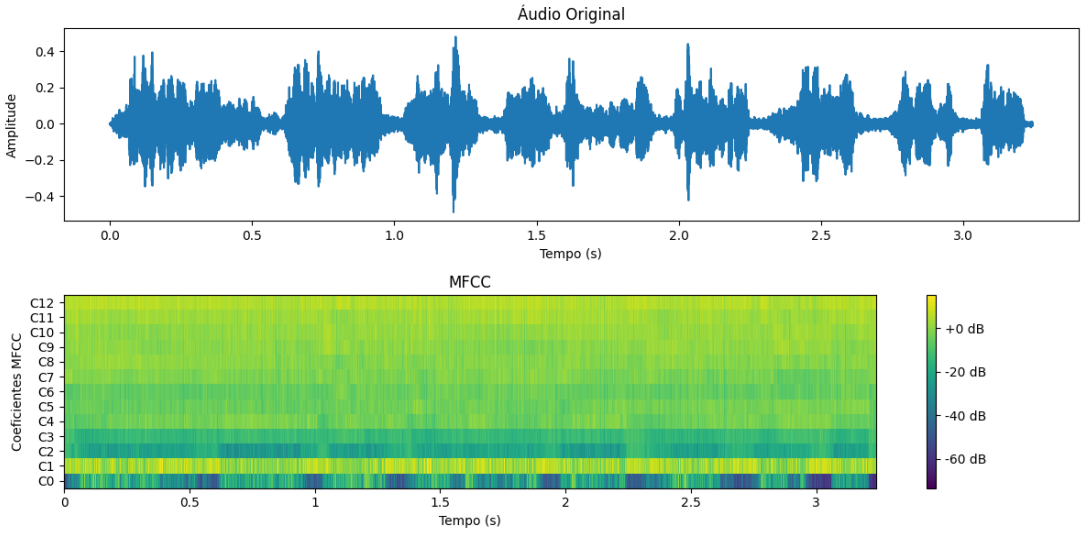
Havia uma função administradora que coordenava a chamada de todos os outros procedimentos, que cumpriam os passos do método, recebendo o vetor da amostra e retornando o vetor de MFCC. A primeira função invocada realizava o janelamento, dividindo o sinal em blocos sobrepostos, conforme descrito na Seção 2.4.4. O resultado era armazenado em uma matriz (vetor de vetores), que servia de entrada para a função seguinte, responsável pelo cálculo do espectro de potência. Nessa etapa, os valores reais da matriz eram convertidos em números complexos ($\mathbb{C}$), que representavam a amplitude e fase da frequência, por meio da aplicação da FFT. Todos os cálculos envolvendo números complexos foram realizados utilizando a biblioteca FFTW3 em C++ (MASSACHUSETTS INSTITUTE OF TECHNOLOGY., 2020), que oferece funções otimizadas para transformadas rápidas de Fourier (FFT).

Esse espectro que era obtido representava a distribuição de energia nas diferentes frequências o sinal, capturando as características espectrais que diferenciavam cada espécie. Os dados eram então armazenados e enviados para uma função que aplicava o banco de filtros de Mel, que convertiam as frequências de Hertz para Mel por meio da Equação 11, apresentada no Capítulo 2. O espectro de cada janela era então multiplicado pelos filtros correspondentes, e calculava-se o logaritmo da energia em cada faixa. O resultado é um vetor de log-energias de Mel, que representa a distribuição espectral do som.

Por fim, o vetor era enviado à função que aplicava a Transformada Discreta do Cosseno (DCT), em que cada coeficiente de característica resultante era calculado de acordo com a Equação 12 do Capítulo 2, concentrando a maior parte da informação relevante nos primeiros coeficientes. Normalmente, os 13 primeiros coeficientes já contêm a maior parte da informação relevante e, portanto, foram os utilizados neste trabalho, como já citado anteriormente.

Na Figura 18, observa-se a distribuição da energia do áudio da arara nos diferentes coeficientes MFCC, em escala logarítmica (dB). Cada ponto indica a intensidade da energia em determinado coeficiente (eixo y) em cada período do áudio (eixo x, em segundos). Nota-se que a arara concentra mais energia em coeficientes intermediários e altos, com concentração também em C1, o que indica características de timbre específicas de sua vocalização. Dessa forma, é possível construir um panorama espectral do áudio e extrair informações discriminativas sobre o canto de cada espécie.

Figura 18 – Aplicação do MFCC no áudio da arara.



Fonte: Produzido pelos autores.

4.3.5 Análise de Componentes Principais (PCA)

Para implementar o PCA em C++, devido ao grau de dificuldade e complexidade computacional dos cálculos matriciais, utilizou-se a biblioteca *Eigen*, uma ferramenta de manipulação de matrizes e operações lineares em C++ (TUXFAMILY, 2025). O PCA foi aplicado no KNN clássico a fins de teste, mas no QKNN era essencial devido limitações de *hardware*.

O fluxo de construção do algoritmo seguiu as etapas descritas no Capítulo 2, sendo eles aplicados da seguinte forma:

1. Cada amostra de áudio foi reorganizada em uma matriz, em que a linha era o áudio observado e a coluna uma componente da suas características. Ou seja, elas foram reorganizadas em uma nova estrutura em C++ que continham as características e a referência do áudio específico observado (linha).

2. Cada componente da *feature* foi centralizado, subtraindo-se a média de sua coluna, de modo que todas as colunas da matriz tivessem média zero. Uma função que calculava a média em C++ era chamada para cada coluna e o cálculo era realizado em cada índice.

3. As *features* também foram normalizadas pelo desvio padrão.

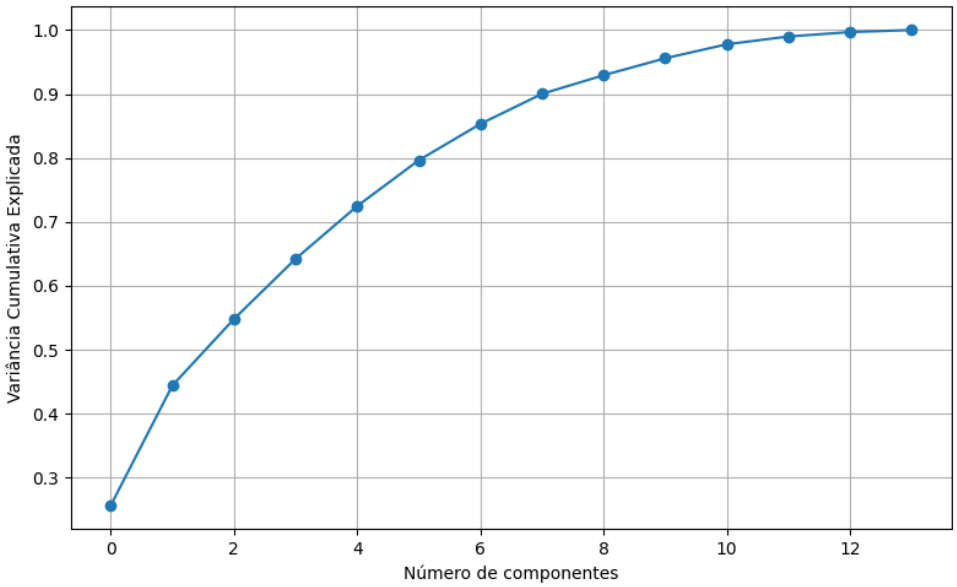
4. A covariância entre todas as *features* foi calculada para analisar a correlação entre elas. Uma função em C++ percorria todos os elementos e correlacionava um a um, aplicando a fórmula de covariância e salvando essa nova matriz.

5. Utilizando a biblioteca *Eigen*, obtiveram-se os autovalores e autovetores da matriz de covariância.

6. Selecionaram-se os primeiros n componentes principais (neste trabalho, foi usado $n = 10$), gerando uma representação reduzida dos dados que preserva a maior parte da variabilidade original.

Na Figura 19, observa-se o gráfico de variância cumulativa, que indica a proporção da variabilidade total dos dados explicada à medida que são adicionados componentes principais (PCs) do PCA. Cada ponto da curva mostra quanto da informação total é retida ao considerar até aquele número de PCs. No caso dos áudios analisados, após a extração de *features* pelo MFCC, por exemplo, cada amostra possuía 14 características, e o gráfico indica que aproximadamente 7 componentes são suficientes para representar cerca de 90% da variância dos dados nesse caso.

Figura 19 – Gráfico da variância cumulativa dos dados (PCA).



Fonte: Produzido pelos autores.

4.4 K-VIZINHOS MAIS PRÓXIMOS (KNN)

Depois da extração de características, foi implementado o KNN utilizando C++. Para isso, criou-se uma estrutura que armazenava o nome da classe e seu vetor de características, que era usado no cálculo de distância entre as amostras. Além disso, foi definido um vetor global para armazenar os resultados de cada comparação durante a classificação de uma espécie, que era limpo a cada nova amostra do conjunto de teste trazida para ser rotulada, garantindo que não houvesse acúmulo de informações entre diferentes classificações.

No código, recebe-se a estrutura do vetor comparado (com rótulo desconhecido) e a do vetor já rotulado, então calculava-se a distância entre os dois (utilizou-se a Euclidiana, de Manhattan e do Cosseno, escolhendo no final a que obteve melhor desempenho) e inseria essa informação no vetor global ordenado de forma crescente por distância (do mais próximo ao mais distante).

Por fim, outra função contava qual classe mais aparecia no vetor, atribuindo-na como rótulo final. Testou-se com diferentes valores de k, como será discutido no Capítulo 5.

Além disso, foi feita a validação cruzada (*K-fold cross-validation*), em que o modelo foi testado em diferentes configurações no conjunto de dados (3, 5, 7 e 9 divisões), com o objetivo de avaliar sua robustez e capacidade de generalização.

Vale ressaltar que durante a metodologia, o KNN foi testado em diferentes configurações e combinações de métricas e hiperparâmetros, de modo a encontrar o conjunto de processos que melhor desempenhasse a tarefa de classificação.

4.5 K-VIZINHOS MAIS PRÓXIMOS QUÂNTICO (QKNN)

Nesta etapa, foi implementada a versão quântica do KNN, o QKNN. Para tal, as etapas de pré-processamento e extração de características usadas foram as mesmas em ambos os modelos, ou seja, o preparo dos dados no QKNN foi feito em paradigma clássico. Com isso, o algoritmo aplicado era híbrido, apenas com o *kernel* quântico.

Vale ressaltar que, para a aplicação do QKNN, foi necessário o uso do PCA, devido às limitações no número de *qubits* disponíveis. No modelo empregado, cada característica dos dados era representada por um *qubit*, o que exigiu a redução da dimensionalidade para adequar o conjunto de atributos à capacidade do sistema quântico.

Ademais, a fórmula de distância implementada foi unicamente a Euclidiana, por ainda ser pouco explorada nesse paradigma.

O QKNN foi implementado em *Python* utilizando o *framework PennyLane* (BERGHOLM et al., 2018), que permite a integração entre algoritmos quânticos e modelos clássicos de ML. Para a IDE, foi usado o *Google Colab*. Como o modelo foi executado em um ambiente de simulação, todo o processamento quântico ocorreu de forma emulada em *hardware* clássico. Isso significa que os estados quânticos foram representados numericamente, em ambiente idealizado e sem a presença de ruídos, decoerência ou erros operacionais típicos de computadores quânticos reais. Essa abordagem oferece estabilidade e controle sobre os experimentos, mas também introduz

limitações, tais quais: a simulação de um circuito quântico exige armazenar explicitamente o estado do sistema no espaço de Hilbert, cuja dimensão cresce exponencialmente com o número de *qubits*. Assim, um sistema com n *qubits* requer a manipulação de 2^n amplitudes complexas na memória, o que rapidamente se torna inviável em *hardware* clássico. Consequentemente, apenas circuitos pequenos podem ser simulados com eficiência, o que justificou a necessidade de aplicar PCA e reduzir o número de atributos do conjunto de dados. Dessa forma, a implementação apresentada deve ser compreendida como uma prova de conceito em ambiente quântico idealizado.

A primeira etapa consistia em codificar a entrada do algoritmo (vetor de características reduzido em n componentes pelo PCA) em estados quânticos. Para isso, cada vetor de entrada $x \in \mathbb{R}^M$ foi transformado em um estado quântico $|\psi(x)\rangle$ a partir de um mapa de características quântico que transcrevia a informação para o espaço de Hilbert ($\mathcal{H}$). Para isso, foram aplicadas rotações no eixo y (R_y) no valor da *feature* x, convertendo-na em um ângulo na esfera de Bloch.

Vale ressaltar que o modelo consistia em um *kernel*, ou seja, uma função que mede a similaridade entre dois pontos no espaço. O *kernel* foi construído utilizando o simulador quântico *lightning.qubit*, que reproduz o comportamento de *qubits* ideais, ou seja, sem ruídos. Ele era definido de acordo com a Equação 2 de similaridade, que mede o grau de sobreposição entre os estados quânticos em x e x'.

$$k(x, x') = |\langle\psi(x)|\psi(x')\rangle|^2, \tag{2}$$

Em outras palavras, o *kernel* avaliava a similaridade entre dois vetores quânticos correspondentes às *features* das amostras. Se o cálculo der 1, os estados são idênticos; se der 0, são ortogonais. O módulo ao quadrado garante que o resultado seja um valor real entre 0 e 1 (sem sinais negativos), que é o grau de similaridade. Essa medida ocorre no espaço de Hilbert de dimensão 2^n, capturando padrões complexos nos dados.

Como o KNN se baseia em medidas de distância, a similaridade é convertida em uma métrica que quantifica o grau de separação entre as amostras, conforme mostrado na Equação 3. A segunda igualdade decorre do fato de que os estados quânticos são normalizados, isto é, $k(x, x) = 1$ para todo x. Assim, quanto menor o valor obtido, maior a similaridade entre os pontos, servindo como critério para a seleção dos k vizinhos mais próximos no QKNN.

$$d(x, x') = \sqrt{k(x, x) + k(x', x') - 2k(x, x')} = \sqrt{2(1 - k(x, x'))}, \tag{3}$$

Na Listagem 4.3, demonstra-se o código que aplica o que foi descrito. Nele, primeiro define-se o número de *qubits*, relacionado à quantidade de características da entrada, ou seja, tem relação com a quantidade de PCs após aplicação do PCA, sendo por isso 10. O *kernel* é construído a partir do *AngleEmbedding*, que transforma os vetores clássicos de entrada em estados quânticos por meio de rotações R_y nos *qubits*, e a partir do *qml.adjoint*, que aplica a operação inversa no segundo vetor, de modo a calcular a sobreposição entre os dois estados

quânticos. Finalmente, a expectativa do projetor é calculada com *qml.expval(qml.Hermitian(...))*, resultando no valor do *kernel* quântico $k(x_1, x_2)$, que representa a similaridade entre os estados codificados. Por fim, a função *dist()* converte essa similaridade em uma métrica de distância real, usada para definir os vizinhos mais próximos.

```
n_qubits = 10
dev_kernel = qml.device("lightning.qubit", wires=n_qubits)
projector = np.zeros((2 ** n_qubits, 2 ** n_qubits))
projector[0, 0] = 1
@qml.qnode(dev_kernel)
def kernel(x1, x2):
    AngleEmbedding(x1, wires=range(n_qubits), rotation="Y")
    qml.adjoint(AngleEmbedding)(x2, wires=range(n_qubits), rotation="
        Y")
    return qml.expval(qml.Hermitian(projector, wires=range(n_qubits))
        )
def dist(x1, x2):
    return np.sqrt(2*(1 - kernel(x1, x2)))
```

Listing 4.3 – Implementação do kernel quântico e distância em Python

5 RESULTADOS

Neste capítulo, são apresentados os resultados em duas partes. Primeiro, os resultados da aplicação método clássico, comparando os hiperparâmetros escolhidos, técnicas de extração de características empregadas, entre outros. Segundo, os resultados da execução do algoritmo quântica, permitindo a comparação entre os paradigmas.

5.1 KNN CLÁSSICO

Para o KNN clássico, serão verificados:

- Escolhas de K;
- Fórmulas de distância;
- Normalizações usadas após a extração de características;
- Métodos de extração de características.

Ao final, serão ainda relatados os resultados considerando diferentes divisões do conjunto de dados, por meio da validação cruzada.

Ressalta-se que, em todas as experimentações, utilizou-se o MFCC como método de extração de características, desde os testes de variação do valor de k até os de normalização. Essa escolha se deve à necessidade de reduzir a dimensionalidade do sinal e o custo computacional do processo de classificação. Além disso, o MFCC foi o método que apresentou o maior ganho isolado de desempenho em relação ao uso direto do sinal bruto, o que será apresentado nos testes com os métodos de extração de características.

Vale salientar que antes da aplicação da validação cruzada, serão apresentados os resultados referentes à partição manual que obteve o melhor desempenho dentre as testadas (70% de dados já rotulados e 30% para serem classificados).

5.1.1 Diferentes valores de K

Primeiro, o KNN foi testado com diferentes valores de k. Para o cálculo das distâncias, foi considerada a média das saídas obtidas usando as três métricas avaliadas (Euclidiana, Cosseno e Manhattan). Dessa forma, os resultados apresentados aqui têm caráter mais geral, usando o MFCC por padrão e sem normalização ainda. O objetivo, portanto, foi apenas determinar preliminarmente o valor mais adequado de k.

Na Tabela 1, observa-se que os valores de k testados são ímpares, uma vez que não foi implementado um método de desempate. Verifica-se que, à medida que k aumenta, o desempenho do modelo é deteriorado, pois a acurácia diminui e o erro cresce, indicando uma redução na

capacidade do classificador de identificar corretamente as classes. Além disso, as métricas de precisão e *recall* revelam que o modelo passa a classificar mais falsos positivos, falhando em reconhecer adequadamente os exemplos pertencentes a cada classe. Esse comportamento é também reforçado pela métrica *F1-score*, que evidencia um maior desequilíbrio entre a capacidade de capturar verdadeiros positivos e evitar falsos positivos com o aumento de k.

Tabela 1 – Métricas estatísticas de avaliação para diferentes valores de k no KNN clássico

K	Acurácia	Macro-Precisão	Macro-F1	Macro-Recall	Erro
1	0,8751	0,8815	0,8749	0,8754	0,1249
3	0,8645	0,8733	0,8628	0,8650	0,1355
5	0,8528	0,8681	0,8535	0,8532	0,1472
7	0,8411	0,8598	0,8423	0,8415	0,1589
9	0,8144	0,8374	0,8139	0,8147	0,1856
11	0,8077	0,8293	0,8072	0,8079	0,1923
15	0,7742	0,8027	0,7727	0,7745	0,2258
19	0,7258	0,7586	0,7235	0,7259	0,2742
21	0,6773	0,7072	0,6737	0,6776	0,3227
33	0,6472	0,6780	0,6418	0,6475	0,3528
45	0,5936	0,6329	0,5867	0,5941	0,4064

Fonte: Elaborado pelos autores.

Em geral, isso indica que, neste conjunto de dados, classificar cada ponto com base apenas no vizinho mais próximo é suficiente para obter bons resultados, sugerindo que há pouca sobreposição entre as classes. Contudo, tal circunstância pode revelar duas situações: ou os dados estão muito bem separados no espaço, o que teria relação com a natureza das classes observadas; ou pode haver um possível sobre-ajuste (*over-fitting*) às amostras usadas, o que comprometeria sua capacidade de generalização. Para o último caso, serão testados outras configurações de dados, na parte de validação cruzada.

5.1.2 Testes com as três fórmulas de cálculo de distância

Nesta etapa, avaliou-se o KNN com três diferentes fórmulas de distância, para o melhor valor de k encontrado na etapa anterior, ou seja, $k = 1$. Na Tabela 2, são apresentadas as métricas estatísticas obtidas em cada caso. Observa-se que a Euclidiana apresentou o pior desempenho, seguida da distância do Cosseno, enquanto a de Manhattan obteve os melhores resultados, apresentando também maior equilíbrio entre a identificação de falsos positivos e a captura correta dos positivos reais por classe (precisão e *recall*), sendo a mais adequada para o problema.

Infere-se, portanto, que as características do conjunto de dados demandam uma medida de distância mais robusta, capaz de lidar com pequenas variações entre as dimensões de cada amostra. Isso sugere a presença de possíveis *outliers* ou distribuições não uniformes entre os atributos, aspectos que são melhor capturados pela distância de Manhattan.

Tabela 2 – Resultados gerais do KNN com diferentes fórmulas de distância.

Distância	Acurácia	Erro	Macro-Precisão	Macro-Recall	Macro-F1
Cosseno	0,8863	0,1137	0,8923	0,8867	0,8849
Euclidiana	0,8378	0,1622	0,8461	0,8380	0,8388
Manhattan	0,9013	0,0986	0,9061	0,9016	0,9011

Fonte: Elaborado pelos autores.

5.1.3 Testes com os diferentes métodos de normalização

Para os experimentos seguintes, adotou-se a distância de Manhattan e $k = 1$, além do MFCC como método de extração de características. As normalizações foram então aplicadas sobre os dados após a etapa de extração com o MFCC, com o objetivo de avaliar se essas técnicas poderiam melhorar a classificação.

Na Tabela 3, observa-se que algumas técnicas de normalização reduziram o desempenho do modelo em comparação à ausência de normalização, como nos casos de *Standard* e *Min-Max*. Por outro lado, as normalizações *Max-Absolute* e *Normalizer* apresentaram resultados semelhantes, com o *Normalizer* sendo levemente melhor. O *Normalizer* aumentou a taxa de acertos por classe, o que sugere que no conjunto de dados as amostras se diferem mais em proporções relativas do que em valores absolutos (magnitude).

Tabela 3 – Resultados gerais do KNN com diferentes técnicas de normalização.

Normalização	Acurácia	Erro	Macro-Precisão	Macro-Recall	Macro-F1
Nenhuma	0,9013	0,0986	0,9061	0,9016	0,9011
Standard	0,8846	0,1154	0,8902	0,8850	0,9013
Min-max	0,8110	0,1889	0,8185	0,8112	0,8110
Max-absolute	0,9047	0,0953	0,9098	0,9050	0,9040
Normalizer	0,9064	0,0936	0,9134	0,9067	0,9058

Fonte: Elaborado pelos autores

É interessante observar que, em relação às métricas de distância, a baseada em direções não foi a que mais beneficiou o modelo; contudo, na etapa de normalização, esse tipo de abordagem apresentou melhor desempenho.

5.1.4 Teste com métodos de extração features

Aqui, as melhores métricas encontradas anteriormente foram empregadas para todos os métodos de extração de características investigados. Os algoritmos de extração de características foram aplicados sozinhos e depois combinados entre si, como apresentado na Tabela 4.

Observa-se que o *Teager*, quando utilizado isoladamente, apresentou desempenho significativamente inferior, com métricas abaixo de 50%, obtendo acurácia de apenas 5,02%. Esse resultado indica que, sozinho, o operador de energia de *Teager* não é eficiente para discriminar as classes. Entretanto, quando combinado com outras técnicas, seu comportamento muda consideravelmente, como no caso de MFCC com *Teager*, que alcançou o melhor desempenho dentre todos

os métodos testados, com acurácia de 98,66%. Isso demonstra que a combinação de métodos complementares, um que captura características espectrais e outro que fornece informações de energia não linear, aumentam a capacidade de identificação do KNN.

Tabela 4 – Métricas do KNN com diferentes métodos de extração de características.

Feature	**Acurácia**	**Erro Rate**	**Macro-Precisão**	**Macro-Recall**	**Macro-F1**
MFCC	0,9064	0,0936	0,9135	0,9067	0,9057
Energia	0,7157	0,2843	0,7320	0,7162	0,7146
ZCR	0,8294	0,1706	0,8720	0,8297	0,8377
Teager	0,0502	0,9498	0,0025	0,0502	0,0048
MFCC + ZCR	0,8293	0,1707	0,8554	0,8296	0,8313
MFCC + *Teager*	0,9866	0,0134	0,9873	0,9867	0,9866
MFCC + Energia	0,9096	0,0903	0,9158	0,9100	0,9090
Energia + ZCR	0,8562	0,1438	0,8850	0,8563	0,8605
Energia + *Teager*	0,7179	0,2821	0,7344	0,7179	0,7176
ZCR + *Teager*	0,5960	0,4040	0,6358	0,5960	0,5979
PCA (n = 10)	0,8681	0,1319	0,8749	0,8680	0,8656

Fonte: Elaborado pelos autores.

O ZCR, quando aplicado de forma isolada, também obteve resultados satisfatórios, com acurácia de 82,94% e macro-*F1* de 83,77%, evidenciando sua boa representatividade das classes. Em combinação com outras features, seu desempenho se manteve consistente: ZCR com Energia acumulada apresentou acurácia de 85,62% e ZCR com MFCC teve acurácia de 82,93%. Contudo, quando combinado com operador de *Teager*, o desempenho caiu um pouco, com acurácia de 59,6% e macro-*F1* de 59,79%. Como ambos atuam no mesmo domínio (temporal) pode ser que eles não acrescentaram informações relevantes em conjunto, reforçando redundâncias.

O método de Energia acumulada, aplicado sozinho, obteve desempenho bom, com acurácia de 71,57%, e apresentou leve melhora quando combinado com ZCR, como supracitado. Entretanto, a combinação de Energia acumulada com *Teager* manteve resultados semelhantes ao uso isolado do primeiro método (acurácia de 71,79% e macro-*F1* de 71,76%), mostrando que o *Teager* não contribuiu de forma significativa neste cenário, possivelmente pela redundância de informação entre ambas as medidas energéticas.

Já a combinação MFCC com Energia resultou em acurácia de 90,96% e macro-*F1* de 90,90%, atingindo um desempenho positivo, com pequena melhora em relação ao uso isolado do MFCC (90,64%). Isso mostra que a energia agrega informação complementar, ainda que não tão expressiva quanto a contribuição observada com o operador de *Teager*.

Por fim, o PCA foi aplicado considerando 10 componentes principais sobre o melhor caso (MFCC + *Teager*), resultando em acurácia de 86,81%, macro-precisão de 87,49% e macro-*F1* de 86,56%. Embora esses valores ainda sejam satisfatórios, observa-se uma queda de 11,85 pontos percentuais em relação a acurácia, indicando que a redução de dimensionalidade provocou perda de informações discriminantes relevantes para o modelo.

Logo, conclui-se que o MFCC em conjunto com o operador de *Teager* é o que permite identificações precisas sobre as classes, indicando que a integração da perspectiva espectral com a energética proporciona representação mais completa das informações acústicas dos cantos das aves.

5.1.5 *K-Cross Fold Validation*

Por fim, apresenta-se os resultados da aplicação da validação cruzada (*K-Fold Cross Validation*), com outros subconjuntos de dados, mas os mesmos parâmetros e métodos anteriormente dados como melhores. Em cada rodada com n divisões, $n - 1$ subdivisões dos dados foram utilizadas como rotulados e uma para ser classificada, garantindo que todas as amostras fossem testadas ao menos uma vez.

A validação cruzada foi aplicada para verificar a presença de sobre-ajuste (*overfitting*) e avaliar a estabilidade do modelo. Na Tabela 5, são apresentados os resultados obtidos com diferentes números de subdivisões (3, 5, 7 e 9), a fim de observar o comportamento do classificador sob diferentes proporções de treino e teste. Observa-se que o desempenho geral manteve-se elevado em todos os casos, com variações sutis entre as métricas. À medida que o número de divisões aumenta, há uma leve melhoria nos resultados.

Tabela 5 – Resultados gerais do KNN com diferentes divisões de pastas.

Divisões	**Acurácia**	**Erro**	**Macro-Precisão**	**Macro-Recall**	**Macro-F1**
3	0,9455	0,0545	0,9462	0,9455	0,9451
5	0,9555	0,0445	0,9559	0,9555	0,9552
7	0,9635	0,0365	0,9637	0,9635	0,9634
9	0,9636	0,0365	0,9637	0,9635	0,9630

Fonte: Elaborado pelos autores.

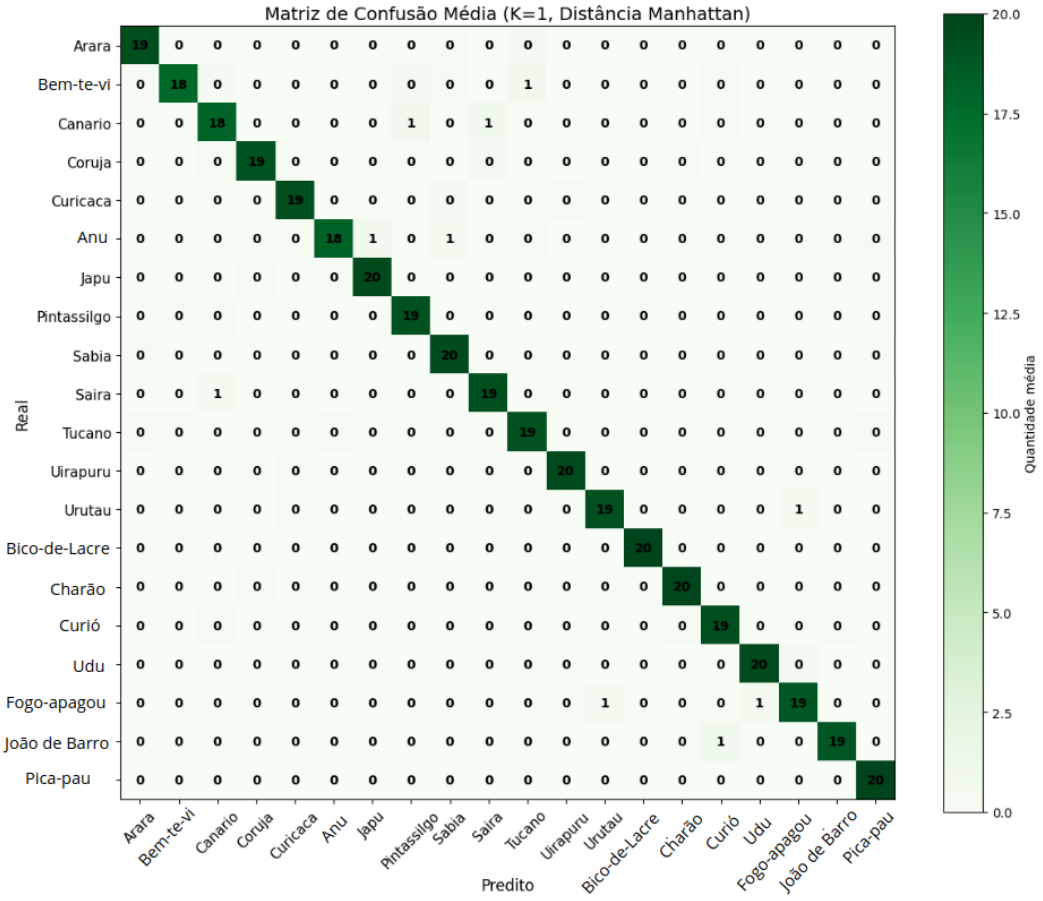
Em comparação com os testes realizados anteriormente, nos quais o conjunto de treino e teste foi definido manualmente, nota-se apenas uma pequena variação nas métricas. A melhor performance foi observada na divisão com nove subconjuntos, embora os resultados com sete divisões tenham sido muito próximos, indicando estabilidade do KNN. As altas precisões refletem que o modelo gerou poucos falsos positivos por classe, enquanto os valores de *recall* mostram que o KNN foi capaz de identificar corretamente a maior parte das amostras em cada categoria.

A validação cruzada oferece uma estimativa mais realista do desempenho do KNN, mitigando a influência de divisões específicas dos dados. Os resultados indicam que o modelo manteve desempenho consistente, sem sinais relevantes de sobre-ajuste, ou seja, ele conseguiu preservar a capacidade de generalização mesmo após múltiplas rodadas de validação.

As métricas apresentadas na Tabela 5 correspondem à média das pontuações obtidas em cada rodada, de modo a resumir o comportamento geral no teste.

Na Figura 20, mostra-se a matriz de confusão média referente à validação com cinco divisões, evidenciando que o modelo classificou corretamente 100% das amostras de doze espécies (arara, coruja, curicaca, japu, pintassilgo, sabiá, tucano, uirapuru, bico-de-lacre, charão, udu-de-coroa-azul e pica-pau), enquanto as demais também apresentaram altas taxas de acerto, mas não 100%. É interessante notar que as espécies em que houve 100% de acerto são as que possuem cantos mais diferentes e característicos em relação às outras.

Figura 20 – Matriz de Confusão média da validação cruzada com 5 pastas.



Fonte: Produzido pelos autores.

Esses resultados reforçam a robustez e a estabilidade do KNN clássico implementado, indicando seu potencial de expansão para um número maior de espécies e gravações, desde que o mesmo processo de pré-processamento e extração de características seja mantido.

5.1.6 Versionamentos em C++ e *Python*

Por fim, apresenta-se uma análise entre duas versões do algoritmo: a implementação original em C++ e uma implementação equivalente em Python utilizando bibliotecas especializadas. O objetivo desta comparação é verificar se a linguagem de programação e o uso de *frameworks* especializados introduziriam diferenças relevantes no comportamento do algoritmo. A escolha

de *Python* deve-se ao fato de ser a linguagem mais utilizada em aplicações de aprendizado de máquina e processamento de sinais, oferecendo suporte por meio de bibliotecas consolidadas.

As comparações subsequentes são realizadas utilizando a combinação de pré-processamento e extração de características que apresentou melhor desempenho, empregando a biblioteca *librosa* para a extração de MFCC, além da variante com PCA, em que utilizou-se a biblioteca *scikit-learn*, que oferece uma implementação consolidada do método.

Na Tabela 6, apresentam-se os resultados obtidos tanto na implementação em C++ quanto na implementação equivalente em *Python*, considerando as seguintes etapas: remoção da componente DC, normalização mínimo–máximo antes da extração de características, extração de MFCC com o operador de Teager, normalização posterior por meio do *normalizer* e classificação utilizando KNN com distância de Manhattan e k=1. Em seguida, compara-se também o desempenho dessas versões quando aplicado o PCA com 10 componentes.

Tabela 6 – Comparação das linguagens *Python* e C++.

Liguagem	**Acurácia**	**Erro Rate**	**Macro-Precisão**	**Macro-Recall**	**Macro-F1**
C++ sem PCA	0,9866	0,0134	0,9873	0,9867	0,9866
Python sem PCA	0,9232	0,0768	0,9274	0,9232	0,9238
C++ com PCA	0,8681	0,1319	0,8749	0,8680	0,8656
Python com PCA	0,9232	0,0768	0,9290	0,9233	0,9233

Fonte: Elaborado pelos autores.

A partir dos resultados apresentados, observa-se que a implementação em C++ obteve o melhor desempenho geral na configuração sem PCA, superando a versão equivalente em *Python* em todas as métricas. Esse resultado indica que a implementação manual em C++ fornece maior controle sobre detalhes numéricos e operacionais do algoritmo, além de menor sobrecarga de execução. Desse modo, há menos generalizações prejudiciais aos resultados do que aquelas introduzidas pelos *frameworks* utilizados no *Python*, o que pode explicar o desempenho superior da versão em C++ nessa configuração.

Por outro lado, quando o PCA com 10 componentes é aplicado, a versão em *Python* passa a apresentar desempenho superior ao C++, possivelmente devido à utilização de bibliotecas mais maduras e otimizadas para operações matriciais e redução de dimensionalidade.

5.2 KNN QUÂNTICO

Nesta etapa, mantiveram-se os passos de pré-processamento aplicadas anteriormente, criando um modelo híbrido clássico-quântico para a resolução do problema. Contudo, algumas modificações foram realizadas: a normalização permaneceu inalterada, mas a métrica de distância utilizada passou a ser a euclidiana, em substituição à de Manhattan, buscando explorar o que foi proposto no artigo apresentado no Capítulo 3. Outra questão é que se manteve o MFCC com operador de *Teager*, mas utilizou-se obrigatoriamente o PCA com 10 componentes, devido

às limitações do simulador quântico no *Google Colab*, que apresentava *timeout* ou estouro de recursos quando um número maior de *qubits* era utilizado.

Ademais, não foi possível aplicar o KNN quântico a todos os áudios simultaneamente devido às mesmas limitações de *hardware* do simulador. Contudo, diferentemente dos modelos clássicos, os algoritmos quânticos não exigem grandes volumes de dados para inferência, sendo capazes de generalizar padrões com um número reduzido de amostras. Ainda assim, para uma comparação mais justa, todos os dados foram utilizados, divididos em subconjuntos aleatórios e sem repetição, que foram processados separadamente.

Inicialmente, os experimentos foram realizados com cinco rodadas aleatórias e balanceadas, número escolhido por representar o primeiro múltiplo que não gerava erros de *timeout* ou estouro de memória no Colab. O processo foi repetido três vezes, calculando-se a média, mediana e desvio padrão das métricas, conforme apresentado na Tabela 7. Posteriormente, foram conduzidos testes com oito e dez rodadas, cujos resultados mostraram estabilidade, motivo pelo qual os experimentos foram encerrados neste ponto.

Tabela 7 – Comparativo das métricas entre experimentos com 5, 8 e 10 rodadas.

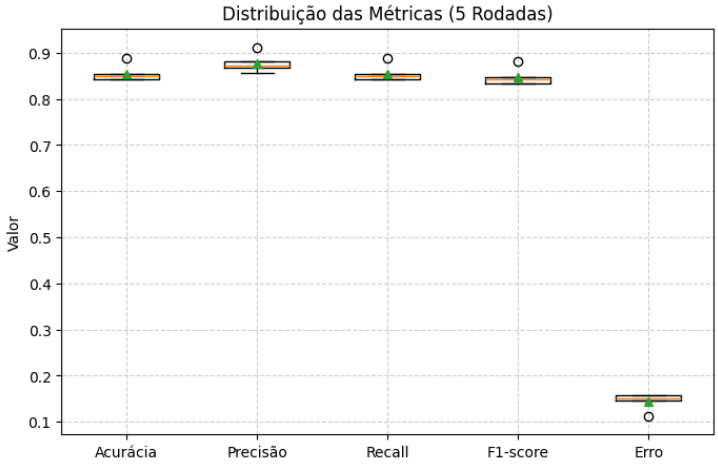
Métrica	**5 Rodadas**			**8 Rodadas**			**10 Rodadas**		
	Média	**Mediana**	**Desvio**	**Média**	**Mediana**	**Desvio**	**Média**	**Mediana**	**Desvio**
Acurácia	0,8550	0,8500	0,0189	0,8359	0,8438	0,0634	0,7950	0,8250	0,0665
Precisão	0,8778	0,8715	0,0208	0,8294	0,8271	0,0763	0,8278	0,8542	0,0676
Recall	0,8550	0,8500	0,0189	0,8302	0,8334	0,0598	0,7950	0,8250	0,0665
F1-score	0,8483	0,8421	0,0200	0,8113	0,8129	0,0693	0,7809	0,8008	0,0699
Erro	0,1450	0,1500	0,0189	0,1641	0,1562	0,0634	0,2050	0,1750	0,0665

Fonte: elaborado pelos autores.

Observa-se que a acurácia diminui conforme o número de rodadas aumenta, indicando uma leve perda na capacidade do modelo em distinguir corretamente as classes com a maior redução do volume de dados. Esse comportamento é acompanhado pelo crescimento do desvio padrão, que sugere redução da estabilidade do QKNN. A precisão manteve-se alta em todos os casos, evidenciando baixo índice de falsos positivos, porém também reduziu ao longo dos testes. Já o *recall* apresentou queda gradual, refletindo menor sensibilidade em detectar todas as amostras positivas por classe nas partições menores. Consequentemente, o *F1-score* também decresceu, revelando uma perda de equilíbrio entre precisão e *recall*. De modo geral, o modelo demonstrou melhor desempenho com cinco rodadas de dados, enquanto números maiores aumentaram a variabilidade e reduziram o desempenho, especialmente no *recall* e no *F1-score*.

Na Figura 21, apresenta-se o desempenho do modelo nas cinco rodadas experimentais. Nota-se a presença de um *outlier* com valores de acurácia, *recall* e *F1-score* próximos de 90%, mas precisão acima de 90%. Isso indica que um dos subconjuntos de dados usados proporcionou condições mais favoráveis para o classificador do que outras.

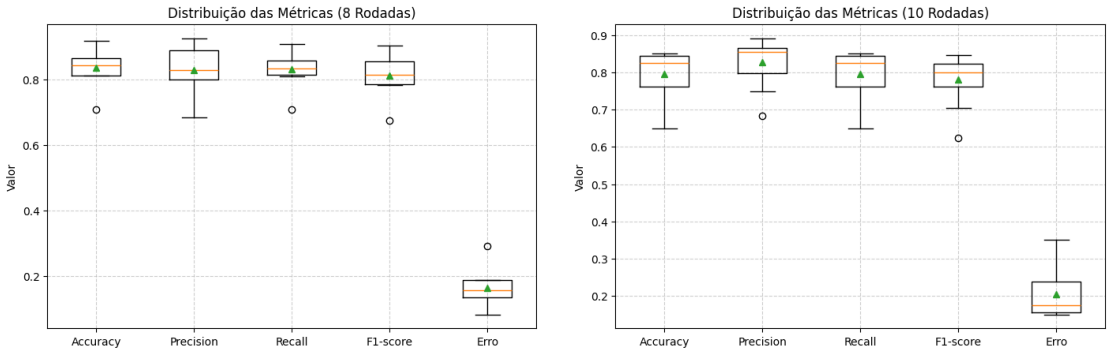
Figura 21 – *Box-plot* das métricas em 5 rodadas do QKNN.



Fonte: Produzido pelos autores.

Na Figura 22, observa-se um comportamento contrário: surgem *outliers* associados a métricas como teve com cinco rodadas, mas com valores inferiores aos demais, o que contribuiu para a redução da média de desempenho apresentados na Tabela 7. Além disso, nota-se que as distribuições das métricas tornam-se menos concentradas, com caixas de *box-plot* mais amplas, refletindo maior dispersão e coerência com o aumento do desvio padrão obtido.

Figura 22 – Comparação dos *box-plots* das métricas do QKNN em diferentes rodadas.



Fonte: Produzido pelos autores.

É importante destacar que essas simulações não consideraram a presença de ruídos quânticos nem técnicas de mitigação deles, fatores relevantes na atual era dos dispositivos quânticos ruidosos de escala intermediária (NISQ). Assim, os resultados apresentados refletem um cenário idealizado, não inteiramente condizente com as limitações práticas da Computação Quântica contemporânea. A única limitação enfrentada foi a quantidade de *qubits* possível de ser implementada, exigindo a divisão dos dados e a aplicação de PCA.

5.3 COMPARAÇÃO: CLÁSSICO X QUÂNTICO

Para garantir uma comparação justa entre o KNN clássico e o KNN quântico, nesta etapa, o modelo clássico foi avaliado com as divisões aleatórias dos subconjuntos de dados (rodadas de 5 pastas, 8 pastas e 10 pastas). No caso do KNN clássico, aplicou-se a redução dimensional por PCA, uma vez que o modelo quântico apresenta restrições quanto ao número de atributos utilizados no processo de codificação quântica, objetivando-se uma comparação mais justa. Dessa forma, os experimentos com o KNN clássico foram realizados utilizando 10 componentes principais. Os resultados estatísticos referentes à acurácia e às demais métricas avaliadas encontram-se na Tabela 8.

Tabela 8 – Experimentos com 5, 8 e 10 rodadas para KNN clássico com PCA de 10 componentes.

Métrica	**5 Rodadas**			**8 Rodadas**			**10 Rodadas**		
	Média	**Mediana**	**Desvio**	**Média**	**Mediana**	**Desvio**	**Média**	**Mediana**	**Desvio**
Acurácia	0,7692	0,8175	0,1062	0,6792	0,6728	0,0614	0,6615	0,6250	0,0922
Precisão	0,8280	0,8390	0,0545	0,7458	0,7497	0,0618	0,6836	0,6700	0,1034
Recall	0,7707	0,8185	0,1050	0,6802	0,6750	0,0614	0,6659	0,6250	0,0896
F1-score	0,7689	0,8118	0,0994	0,6738	0,6765	0,0625	0,6371	0,6152	0,1018
Erro	0,2308	0,1825	0,1062	0,3208	0,3272	0,0614	0,3385	0,3750	0,0922

Fonte: elaborado pelos autores.

Ao comparar os resultados com os obtidos pelo KNN quântico (Tabela 7), observa-se que o modelo quântico apresentou desempenho superior em todas as configurações de rodadas. Em média, o QKNN obteve maior acurácia (0,8550; 0,8359; 0,7950) em relação ao KNN (0,7692; 0,6792; 0,6615), além de valores mais altos de precisão, *recall* e *F1-score*. Outro destaque é o menor desvio-padrão do modelo quântico, indicando maior estabilidade em algumas configurações. Assim, os resultados sugerem que o QKNN foi mais robusto e consistente, especialmente em cenários com maior quantidade de subdivisões e menos dados, mesmo operando com a limitação dimensional que motivou o uso do PCA no modelo clássico para essa comparação.

6 CONCLUSÃO

Os experimentos realizados com o KNN, em suas versões clássica e quântica, evidenciam que a combinação adequada entre técnicas de pré-processamento, extração de características e escolha de hiperparâmetros exerce impacto direto no desempenho do modelo. Embora seja um método simples, o KNN pode alcançar resultados competitivos quando os dados são tratados adequadamente, comparáveis ao de modelos mais complexos.

Quando aplicado ao conjunto completo de dados, o KNN clássico teve o melhor desempenho geral, alcançando acurácia de 98,66%, evidenciando elevada capacidade discriminativa quando dispõe de um número maior de amostras representativas. Contudo, ao analisar os experimentos com subdivisões dos dados (5, 8 e 10 rodadas), o QKNN apresentou-se mais robusto. Nessas condições, o KNN clássico sofreu uma queda acentuada no desempenho, evidenciando maior dependência de conjuntos maiores para manter sua capacidade discriminativa. Em contraste, o QKNN apresentou reduções mais suaves com a diminuição do tamanho dos conjuntos de dados e chegou a obter, em uma das rodadas, acurácia próxima a 90%, demonstrando maior estabilidade com menos amostras. Esse resultado sugere que o modelo quântico explora de forma mais eficiente a estrutura representacional reduzida no espaço de Hilbert, tornando-se menos sensível à variação dos subconjuntos.

Em suma, o KNN clássico destaca-se por sua escalabilidade, simplicidade e desempenho quando operado com quantidades grandes de dados e maior número de atributos. O QKNN, embora limitado pelo *hardware* e pelo número de componentes, mostra-se competitivo em cenários restritos. Assim, com o avanço do *hardware* quântico e de técnicas de codificação, o QKNN tende a superar gradualmente suas limitações atuais, tornando-se cada vez mais promissor frente aos modelos clássicos em aplicações reais.

Para trabalhos futuros, propõe-se:

- Ampliar o número de classes, incluindo novas espécies;
- Testes com conjuntos de dados maiores, especialmente para o modelo clássico;
- Garantir que o classificador identifique corretamente a espécie independentemente do tipo de canto emitido, já que essa variação intra-espécie aumenta a diversidade dos padrões acústicos e dificulta a separação das classes.
- Buscar aprimorar o modelo quântico, tornando-o mais generalista e escalável;
- Executar testes em diferentes *hardwares* quânticos e considerar a presença de ruídos reais;
- Investigar como aprimorar o pré-processamento quando há múltiplas aves de diferentes espécies cantando simultaneamente, além de meios de amenizar ruídos sonoros ambientais.

REFERÊNCIAS

AIMEUR, E.; BRASSARD, G.; GAMBS, S. Quantum speed-up for unsupervised learning. **Machine Learning**, v. 90, p. 261–287, 2013.

AMORIM, L. B. d.; CAVALCANTI, G. D.; CRUZ, R. M. The choice of scaling technique matters for classification performance. **Applied Soft Computing**, v. 133, p. 109924, 2023. ISSN 1568-4946.

BANG, A. V.; REGE, P. P. Classification of bird species based on bioacoustics. In: **International Conference on Advances in Computer and Information Technology ACIT 2013**. Kuala Lumpur , Malaysia: ACIT, 2013. p. 33–37.

BANG, A. V.; REGE, P. P. Recognition of bird species from their sounds using data reduction techniques. In: **7th International Conference on Computer and Communication Technology**. New York, NY, USA: ACM, 2017.

BAPTISTA, L. F.; GAUNT, S. L. L. Bioacoustics as a tool in conservation studies. In: CAMBRIDGE UNIVERSITY PRESS. **Behavioral Approaches to Conservation in the Wil**. 1. ed. Cambridge: Cambridge University Press, 1997. p. 212–242.

BERGHOLM, V. et al. **PennyLane**. USA, 2018. Disponível em: https://arxiv.org/abs/1811.04968. Acesso em: 2 nov. 2025.

CAO, Y. et al. Quantum chemistry in the age of quantum computing. **Chemical Reviews**, v. 119, n. 19, p. 10856–10915, 2018.

CHAKRAVORTY, P. What is a signal? **IEEE Signal Processing Magazine**, v. 35, n. 5, p. 175–177, 2018.

CHANDRAMOULI, S.; DUTT, S.; DAS, A. K. **Machine Learning**. India: Pearson, 2019.

CHOWDHURY, D. R.; NATH, A. Introduction to quantum gates: A comprehensive study on quantum information. **International Journal of Innovative Research in Technology (IJIRT)**, v. 9, n. 6, p. 433–441, 2022.

DAVIS, S. B.; MERMELSTEIN, P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. **IEEE Transactions on Acoustics, Speech, and Signal Processing**, v. 28, n. 4, p. 357–366, 1980.

DEEP LEARNING UNIVERSITY. **Qiskit RY Gate**. [S.l.], 2025. Imagem digital, formato png. Disponível em: https://deeplearninguniversity.com/qiskit/qiskit-ry-gate/. Acesso em: 2 nov. 2025.

DIRAC, P. A. M. **The Principles of Quantum Mechanics**. 4th. ed. Oxford: Clarendon Press, 1958.

GARTNER RESEARCH. **Market Share: Enterprise Software, Worldwide, 2024**. Gartner Research, 2024. Disponível em: https://www.gartner.com/en/documents/6372011. Acesso em: 27 out. 2025.

GIBB, R. et al. Emerging opportunities and challenges for passive acoustics in ecological assessment and monitoring. **Methods in Ecology and Evolution**, v. 10, n. 2, p. 169–185, 2018.

GOODALL, J.; BERMAN, P. **Reason for Hope: A Spiritual Journey**. 1. ed. [S.l.]: Grand Central Publishing, 1999. ISBN 0446676136.

GORRIZ, J. et al. **Is K-fold cross validation the best model selection method for Machine Learning?** arXiv, 2024. Disponível em: https://arxiv.org/abs/2401.16407. Acesso em: 29 out. 2025.

GREENFIELD, P. **More than half of world's bird species in decline, as leaders meet on extinction crisis**. 2025. Disponível em: https://www.theguardian.com/environment/2025/oct/10/biodiversity-conservation-world-bird-species-declining-arctic-seals-green-turtles-iucn-uae-aoe. Acesso em: 30 out. 2025.

GRUS, J. **Data Science from Scratch**. Sebastopol, CA, USA: O'Reilly Media, Inc., 2015. ISBN 978-1-491-90142-7.

GUIDO, R. C. A tutorial on signal energy and its applications. **Neurocomputing**, v. 264, p. 264–282, 2016a.

GUIDO, R. C. Zcr-aided neurocomputing: A study with applications. **Knowledge-Based Systems**, v. 105, p. 248–269, 2016b.

GUPTA, S. et al. Feature extraction using mfcc. **Signal Image Processing : An International Journal (SIPIJ)**, v. 4, n. 4, p. 101–108, 2013.

GÉRON, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow**. Sebastopol, CA, USA: O'Reilly Media, Inc., 2019.

HAMSICI, O. C.; MARTINEZ, A. M. Spherical-homoscedastic distributions: The equivalency of spherical and normal distributions in classification. **Journal of Machine Learning Research**, v. 8, n. 56, p. 1583–1623, 2007.

HOPER, D. U. et al. A global synthesis reveals biodiversity loss as a major driver of ecosystem change. **Nature**, v. 486, p. 105–108, 2012.

INSTITUTO NACIONAL DE METEOROLOGIA. **El Niño: saiba como foi a atuação do fenômeno no Brasil**. INMET, 2025. Disponível em: https://portal.inmet.gov.br/noticias/el-ni%C3%B1o-saiba-como-foi-a-atua%C3%A7%C3%A3o-do-fen%C3%B4meno-no-brasil. Acesso em: 27 jul. 2025.

JACKSON, J. E. **A user's guide to principal components**. Nova York, USA: Wiley, 1991.

JOLLIFFE, I. T.; CADIMA, J. Principal component analysis: a review and recent developments. **Philosophical Transactions of The Royal Society A**, v. 374, n. 2065, 2016.

KAISER, J. On a simple algorithm to calculate the 'energy' of a signal. In: IEEE, 1990. **International Conference on Acoustics, Speech, and Signal Processing**. Albuquerque, NM, USA: IEEE, 1990. v. 1, p. 381–384.

KHRENNIKOV, A. Roots of quantum computing supremacy: superposition, entanglement, or complementarity? **European Physical Journal Special Topics**, v. 230, p. 1053–1057, 2021.

LUCA, G. D. A survey of nisq era hybrid quantum-classical machine learning research. **Journal of Artificial Intelligence and Technology**, v. 2, n. 1, 2022.

MADHUSUDHANA, S.; GAVRILOV, A.; ERBE, C. Automatic detection of echolocation clicks based on a gabor model of their waveform. **Journal of the Acoustical Society of America**, v. 137, n. 6, p. 3077–3086, 2015.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY. **FFTW**: Version 3.3.10. Massachusetts, 2020. Disponível em: https://www.fftw.org/fftw3.pdf. Acesso em: 1 nov. 2025.

MCFEE, B. et al. librosa: Audio and music signal analysis in python. In: SCIPY. **Proceedings of the 14th Python in Science Conference**. 2015. p. 18–25. Disponível em: https://zenodo.org/badge/6309729.svg.

MCKINSEY. **The state of AI in early 2024: Gen AI adoption spikes and starts to generate value**. McKinsey, 2024. Disponível em: https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai-2024. Acesso em: 27 out. 2025.

MEKONEN, S. Birds as biodiversity and environmental indicator. **Advances in Life Science and Technology**, v. 60, p. 16–22, 2017.

MENDEZ, M. A. Continuous and discrete lti systems. In: CAMBRIDGE UNIVERSITY PRESS. **Data-Driven Fluid Mechanics: Combining First Principles and Machine Learning**. 1. ed. Cambridge: Cambridge University Press, 2023. p. 61–88.

MICROSOFT. **Resource Interchange File Format (RIFF)**. Redmond, WA, USA: Microsoft, 2025. Disponível em: https://learn.microsoft.com/en-us/windows/win32/xaudio2/resource-interchange-file-format--riff-. Acesso em: 1 nov. 2025.

MINOCHA, S.; NAMASUDRA, S. Chapter ten - research challenges and future work directions in dna computing. **Advances in Computers**, v. 129, p. 363–387, 2023.

MUTANU, L. et al. A review of automated bioacoustics and general acoustics classification research. **Sensors**, v. 22, n. 21, p. 8361, 2022.

NAMMOUCHI, A.; KASSLER, A.; THEORACHIS, A. Quantum machine learning in climate change and sustainability: a review. **Applied Sciences**, v. 15, n. 8, p. 4119, 2025.

NASTESKI, V. An overview of the supervised machine learning methods. **HORIZONS B**, v. 4, p. 51–62, 2017.

NETO, J. S. et al. **Aprendizado de Máquina Quântica: Teoria e Aplicações**. 1. ed. [S.l.]: CBNA, 2025. ISBN 978-65-01-33992-4.

NIELSEN, M. A.; CHUANG, I. L. **Quantum Computation and Quantum Information**. United Kingdom at the University Press, Cambridge: Cambridge University Press, 2010.

OPPENHEIM, A. V.; SCHAFER, R. W. **Discrete-Time Signal Processing**. 3. ed. Upper Saddle River, NJ: Pearson, 2010.

PACHECO, J. F. et al. Lista comentada das aves do brasil pelo comitê brasileiro de registros ornitológicos. **Ornithology Research**, v. 29, n. 2, p. 2–123, 2021.

PATTRAKSHA, A. A. G. R. K. et al. Classification of bird sounds using the mel-frequency cepstral coefficient (mfcc) and k-nearest neighbor (knn) methods. **JELIKU (Jurnal Elektronik Ilmu Komputer Udayana)**, v. 11, n. 3, p. 679, 2023.

PINHEIRO, B. et al. Teager energy operator as a metric to evaluate local synchronization of power system devices. **Sustainable Energy, Grids and Networks**, v. 43, p. 101853, 2025. ISSN 2352-4677.

PRESKILL, J. Quantum computing in the nisq era and beyond. **Quantum**, v. 2, p. 79–99, 2018.

REZAEI, T.; JAVADI, A. Environmental impact assessment of ocean energy converters using quantum machine learning. **Journal of Environmental Management**, v. 362, p. 121–275, 2024.

RODRIGUES, J. et al. A robust dual-mode machine learning framework for classifying deforestation patterns in amazon native lands. **Land**, v. 13, n. 09, p. 1427, 2024.

ROSA, T.; PRIMARTHA, R.; WIJAYA, A. Comparison of distance measurement methods on k-nearest neighbor algorithm for classificationg. In: SICONIAN, 2019. **Sriwijaya International Conference on Information Technology and Its Applications (SICONIAN)**. [S.l.]: Atlantis Press SARL, 2019. p. 358–361.

SANATI, A.; BORZOEI, O. **Concise Insights into Quantum Machine Learning and Its Practical Uses**. 2024. Disponível em: https://www.researchgate.net/publication/384196939_Concise_Insights_into_Quantum_Machine_Learning_and_Its_Practical_Uses. Acesso em: 8 set. 2025.

SANTOS, V. O. et al. Optimizing the architecture of a quantum–classical hybrid machine learning model for forecasting ozone concentrations: Air quality management tool for houston, texas. **Atmosphere**, v. 16, n. 3, p. 255, 2025.

SELFRIDGE, O. G. Pattern recognition and modern computers. In: WESTERN JOINT COMPUTER CONFERENCE, 1955. **Proceedings of the March 1-3**. New York, NY, USA: Association for Computing Machinery, 1955. p. 91–13.

SHARMA, S.; SATO, K.; GAUTAM, B. P. A methodological literature review of acoustic wildlife monitoring using artificial intelligence tools and techniques. **Sustainability**, v. 15, n. 9, p. 7128, 2023.

SHI, T. et al. Memristor-based feature learning for pattern classification. **Nature Communications**, v. 16, p. 913, 2025.

SOKOLOVA, M. S.; LAPALME, G. A systematic analysis of performance measures for classification tasks. **Information Processing and Management**, v. 45, n. 4, p. 427–437, 2009.

SUYAL, M. S.; GOYA, P. G. A review on analysis of k-nearest neighbor classification machine learning algorithms based on supervised learning. **International Journal of Engineering Trends and Technology**, v. 70, n. 7, p. 43–48, 2022.

TUBARO, P. L. Bioacústica aplicada a la sistemática, conservación y manejo de poblaciones naturales de aves. **Etología**, Buenos Aires, Argentina, v. 7, p. 19–32, 1999.

TUXFAMILY. **Eigen is a C++ template library for linear algebra**. USA, 2025. Disponível em: https://eigen.tuxfamily.org/index.php?title=Main_Page. Acesso em: 2 nov. 2025.

WILMOTT, P. **Machine Learning: An Applied Mathematics Introduction**. 1. ed. Ghana: Panda Ghana Publishing, 2019. ISBN 978-1-9160816-0-4.

WORLD METEOROLOGICAL ORGANIZATION. **State of the Climate in Latin America and the Caribbean 2023**. WMO, 2024. Disponível em: https://wmo.int/publication-series/state-of-climate-latin-america-and-caribbean-2023. Acesso em: 27 jul. 2025.

WORLD RESOURCES INSTITUTE. **Global Forest Loss Shatters Records in 2024, Fueled by Massive Fires**. WRI, 2025. Disponível em: https://www.wri.org/news/release-global-forest-loss-shatters-records-2024-fueled-massive-fires. Acesso em: 27 jul. 2025.

WU, C. S.; KOSURU, S.; TIPPAREDDY, S. Bird species identification from audio data. In: **Proceedings of the 2023 IEEE Ninth International Conference on Big Data Computing Service and Applications (BigDataService)**. Athens, Greece: IEEE, 2023.

XENO-CANTO. **Xeno-Canto**. 2025. Disponível em: https://www.xeno-canto.org/. Acesso em: 2 nov. 2025.

ZARDINI, E.; BLANZIERI, E.; PASTORELLO, D. A quantum k-nearest neighbors algorithm based on the euclidean distance estimation. **Quantum Machine Intelligence**, v. 6, p. 23, 2024.

ZAVALA, F. **Fundamentals of Image Processing in the Frequency Domain**. [S.l.], 2025. Imagem digital, formato png. Disponível em: https://medium.com/imagecraft/fundamentals-of-image-processing-in-the-frequency-domain-ce9ec830181d. Acesso em: 1 set. 2025.

GLOSSÁRIO

Clusterização: é uma técnica de aprendizado não supervisionado que agrupa dados semelhantes em conjuntos chamados *clusters*.

***Pipeline*:** é uma sequência de etapas organizadas para processar dados de forma automática e ordenada.

Metadados: são informações estruturadas que descrevem um conjunto de dados, funcionando como um meio de contextualizar e documentar eles.

***Outliers*:** dados anômalos que se afastam do comportamento geral do conjunto.

***Timeout*:** situação em que um processo ou operação é interrompido automaticamente porque excedeu o tempo máximo permitido para execução.

DADOS CURRICULARES

IDENTIFICAÇÃO	
	Júlia Rodrigues Marques do Nascimento 06/01/2002
Nacionalidade	Brasileira
Nome em citações bibliográficas	NASCIMENTO, J. R. M.; RODRIGUES, JULIA.
Currículo Lattes	http://lattes.cnpq.br/3644416916616249
ORCID	https://orcid.org/0000-0002-0368-7536
FORMAÇÃO ACADÊMICA	
2022/atual	Bacharelado em Ciência da Computação Universidade Estadual Paulista (UNESP), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto
PRODUÇÃO BIBLIOGRÁFICA	
RODRIGUES, JULIA; DIAS, MAURICIO ARAÚJO ; NEGRI, ROGÉRIO ; HUSSAIN, SARDAR MUHAMMAD ; CASACA, WALLACE . A Robust Dual-Mode Machine Learning Framework for Classifying Deforestation Patterns in Amazon Native Lands. LAND, v. 13, p. 1427, 2024.	
PARTICIPAÇÃO EM EVENTOS CIENTÍFICOS	
CIC, 2023, (São José do Rio Preto). ANÁLISE DE DESMATAMENTO DA AMAZÔNIA POR MEIO DE ALGORITMOS DE DETECÇÃO DE ANOMALIAS. 2023. (Congresso). CIC, 2024, (São José do Rio Preto). EXPLORANDO TÉCNICAS DE DETECÇÃO DE ANOMALIAS E REDES NEURAIS PROFUNDAS PARA CLASSIFICAÇÃO DE DESMATAMENTO NA AMAZÔNIA BRASILEIRA. 2024. (Congresso).	