UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
DEPARTAMENTO DE COMPUTAÇÃO
ENGENHARIA DE COMPUTAÇÃO

Natanael Monteiro Pastore Antonioli

Análise do desempenho de diferentes vetorizadores e diferentes classificadores na identificação de espécies de anuros através do som

São Carlos
2025

Natanael Monteiro Pastore Antonioli

Análise do desempenho de diferentes vetorizadores e diferentes classificadores na identificação de espécies de anuros através do som

Trabalho de Conclusão de Curso de Graduação em Engenharia de Computação do Centro de Ciências Exatas e Tecnológicas da Universidade Federal de São Carlos como requisito para a obtenção do título de Engenheiro de Computação.

Orientador: Prof. Dr. Murilo Coelho Naldi

São Carlos
2025

Antonioli, Natanael Monteiro Pastore

Análise do desempenho de diferentes vetorizadores e diferentes classificadores na identificação de espécies de anuros através do som / Natanael Monteiro Pastore Antonioli -- 2025.
42f.

TCC (Graduação) - Universidade Federal de São Carlos, campus São Carlos, São Carlos Orientador (a): Murilo Coelho Naldi. Banca Examinadora: Heloisa de Arruda Camargo, Cesar Henrique Comin
Bibliografia

1. Classificação de som. 2. Vetorização de som. 3. Anuro. I. Antonioli, Natanael Monteiro Pastore. II. Título.

Ficha catalográfica desenvolvida pela Secretaria Geral de Informática (SIn)

DADOS FORNECIDOS PELO AUTOR

Bibliotecário responsável: Arildo Martins - CRB/8 7180

Natanael Monteiro Pastore Antonioli

Análise do desempenho de diferentes vetorizadores e diferentes classificadores na identificação de espécies de anuros através do som

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Engenheiro de Computação" e aprovado em sua forma final pelo Curso de Graduação em Engenharia de Computação.

São Carlos, 21 de Fevereiro de 2025.

Banca Examinadora:

_____

Prof. Dr. Murilo Coelho Naldi
Universidade Federal de São Carlos

_____

Profa. Dra. Heloisa de Arruda Camargo
Universidade Federal de São Carlos

_____

Prof. Dr. Cesar Henrique Comin
Universidade Federal de São Carlos

RESUMO

Anfíbios sem cauda, conhecidos como anuros, são um grupo diverso que inclui sapos, rãs e pererecas e que, quando monitorados, fornecem ótimos indicadores da degradação de um ecossistema e da presença de espécies invasivas. O presente trabalho determina a melhor combinação na forma de um par composto por algoritmo de vetorização e algoritmo de classificação na tarefa de identificação de espécies de anuros através de gravações sonoras. O estudo tem como objetivo principal determinar valores de acurácia, precisão e revocação de cinco algoritmos de vetorização combinados com seis algoritmos de classificação. Para tanto, esses algoritmos foram implementados em Python e iterados em cada combinação possível, com as métricas de avaliação calculada. Os resultados demonstram que transformada de Fourier de curto termo e cepstrum de frequência Mel apresentam ótimos resultados quando combinados com *perceptron* multicamadas e com floresta aleatória. Conclui-se que existem diversos métodos promissores para identificação de anuros através do som, e que estes métodos podem ser explorados e escolhidos conforme as restrições e necessidades da pesquisa na qual serão aplicados.

Palavras-chave: Classificação de som; Vetorização de som; Anuros; Aprendizado de máquina.

ABSTRACT

Tail-less amphibians, known as anurans, are a diverse group that includes toads, frogs, and tree frogs, and when monitored, they serve as excellent indicators of ecosystem degradation and the presence of invasive species. This study determines the best combination in the form of a pair composed of a vectorization algorithm and a classification algorithm for the task of identifying anuran species through audio recordings. The main objective of the study is to determine accuracy, precision, and recall values for five vectorization algorithms combined with six classification algorithms. To this end, these algorithms were implemented in Python and iterated over every possible combination, with evaluation metrics calculated. The results show that short-time Fourier transform and mel-frequency cepstrum yield excellent results when combined with multilayer perceptron and random forest. It is concluded that there are several promising methods for identifying anurans through sound, and that these methods can be explored and selected according to the constraints and requirements of the research in which they are to be applied.

Keywords: Sound classification; Sound vectorization; Anurans; Machine learning.

AGRADECIMENTOS

Aos meus pais, Valéria e Gilberto, que não me repreenderam quando, duas décadas atrás, me interessei por sapos e, por mais vezes que consigo contar, achei que seria uma boa ideia entrar em casa segurando um ou mais sapos.

À minha namorada, Laura, que me apoiou durante o desenvolvimento deste trabalho de conclusão de curso e me mostrou que, independentemente das dificuldades enfrentadas, tudo daria certo ao final.

Aos meus amigos, Henrique, Luiz, Vitor e Gabriel, que me ouviram falar sobre sapos por anos e, gradualmente, me ajudaram a formular a ideia para esse trabalho de conclusão de curso.

Aos meus professores, Murilo e Heloisa, que me apresentaram ao mundo da inteligência artificial e me guiaram ao longo das disciplinas obrigatórias e optativas.

"Na ciência experimental, a natureza é a nossa melhor amiga e melhor crítica se permitirmos que suas sugestões sejam imparciais em nossas mentes,"

Michael Faraday

LISTA DE FIGURAS

Figura 1 – etapas envolvidas. Fonte: elaborado pelo autor..... 3
Figura 2 – módulo de captação de áudio. Fonte: (CHAO et al., 2019). 4
Figura 3 – filtro de mel. Fonte: (A. MAJEED et al., 2015)..... 8
Figura 4 – exemplo de hiperplano em máquina de vetores de suporte. Fonte: ("SVM-Kernels", [s.d.]). 13
Figura 5 – Matriz de confusão com quatro classes a partir da classe B. Fonte: elaborado pelo autor, adaptado de (RAINIO; TEUHO; KLÉN, 2024). 21

LISTA DE TABELAS

Tabela 1 – códigos associados aos nomes científicos e vulgares de cada anfíbio, bem como o total de arquivos e a duração total. Fonte: (AKBAL et al., 2023), adaptado pelo autor. 18
Tabela 2 – resultado da métrica de precisão. 23
Tabela 3 – resultado da métrica de acurácia. 24
Tabela 4 – resultado da métrica de revocação. 24
Tabela 5 – aplicação da escala nos resultados obtidos 25

LISTA DE ABREVIATURAS E SIGLAS

AAC	advanced audio coding
CPU	central processing unit
GPU	graphics processing unit
kNN	k-nearest neighbor
M4A	MPEG-4 Audio Layer
MFC	mel-frequency cepstrum
MFCCs	mel-frequency cepstral coefficients
SVM	support vector machine
STFT	short-time Fourier transform
QDA	quadrant discriminant analysis
ReLU	rectified linear unit

SUMÁRIO

1 Introdução 1

1.1 OBJETIVOS 2

1.1.1 Objetivo geral 2

1.1.2 Objetivos específicos 2

2 Fundamentação teórica 3

2.1 REVISÃO DA LITERATURA 3

2.2 ETAPAS PARA A CONSTRUÇÃO DE UM SISTEMA 3

2.2.1 Coleta 4

2.2.2 Rotulagem 4

2.2.3 Vetorização 4

2.2.4 Treinamento e teste 5

2.2.5 Aplicação 5

2.3 ARQUIVOS DE ÁUDIO 6

2.4 ALGORITMOS DE VETORIZAÇÃO 6

2.4.1 Transformada curta de Fourier 6

2.4.2 MFCCs 7

2.4.3 Rastreio de altura 9

2.4.4 Croma 10

2.4.5 Energia 11

2.5 ALGORITMOS DE CLASSIFICAÇÃO 11

2.5.1 Bayes ingênuo 12

2.5.2 Máquina de vetores de suporte 12

2.5.3 *k*-vizinhos mais próximos 13

2.5.4 Análise de discriminante quadrático 14

2.5.5 *Perceptron* multicamadas 14

2.5.6 Floresta aleatória 16

3 Materiais e métodos 17
3.1 FERRAMENTAS 17
3.1.1 Linguagem e bibliotecas 17
3.1.2 Base de dados 18
3.1.3 Arquitetura do projeto 19
4 Análise experimental e discussão 21
4.1 CONFIGURAÇÃO EXPERIMENTAL 21
4.2 MÉTRICAS DE AVALIAÇÃO 21
4.2.1 Matriz de confusão 21
4.2.2 Acurácia 22
4.2.3 Precisão 23
4.2.4 Revocação 23
4.3 RESULTADOS 23
4.4 DISCUSSÃO DOS RESULTADOS 24
5 Conclusão 26
5.1 LIMITAÇÕES 26
5.2 TRABALHOS FUTUROS 26
5.3 CONSIDERAÇÕES FINAIS 27

1 INTRODUÇÃO

Anuros são anfíbios que compreendem a ordem *Anura*, composta de sapos, rãs e pererecas. Anuros possuem uma pele permeável, e esse órgão desempenha um papel importante na respiração e na hidratação do indivíduo. Justamente por essa razão, anuros são extremamente vulneráveis a contaminantes, o que faz com que a presença ou ausência de indivíduos dessa ordem seja um marcador da qualidade de um ecossistema (JOSEPH MUSONDA KABANZE et al., 2023).

Além disso, alguns anfíbios vivem em certas regiões como espécies invasoras: esse é o caso do sapo cururu (*Rhinella marina*), importado para a Austrália em 1935 para combater um besouro da cana-de-açúcar (DEPARTMENT OF THE ENVIRONMENT, WATER, HERITAGE AND THE ARTS, 2021). Com as tendências de mudanças climáticas observadas atualmente, estima-se que os habitats disponíveis para consideradas quatro piores espécies invasoras de anuros irá aumentar consideravelmente (ANDERSEN; BORZÉE; JANG, 2021).

No Brasil, um país com grande atividade agrícola e com parte considerável da população habitando zonas rurais e dependendo da água extraída de córregos ou poços, o monitoramento da população de anuros é um parâmetro importante para estudo da qualidade do ecossistema e para mitigar problemas que venham a comprometê-la antes que essa população seja mais gravemente acometida.

Como quase todos os anuros emitem um som característico da espécie para atrair fêmeas durante períodos de acasalamento (STARNBERGER; PREININGER; HÖDL, 2014), monitorar sons de anuros tem se mostrado uma excelente forma de monitorar a qualidade de um ecossistema, bem como identificar eventuais espécies invasoras.

Para realizar essa tarefa, a maioria das propostas procura identificar a presença de anuros através do som gravado por um aparelho. Inicialmente, biólogos especializados são empregados para utilizar sua expertise na identificação, mas esse é um processo caro e demorado. Usando aprendizado de máquina, é possível utilizar gravações previamente identificadas por biólogos para treinar um algoritmo responsável por realizar futuras identificações em tempo muito mais curto e com custo muito menor.

Visando atingir esse objetivo, é preciso escolher um algoritmo de vetorização, responsável por converter o arquivo de áudio em um conjunto de números de acordo com

uma característica; e um algoritmo de classificação, que será treinado usando uma fração n do conjunto de áudios disponíveis, e testado usando uma fração $1 - n$ do mesmo conjunto.

Este projeto propõe testar diversos algoritmos de vetorização exaustivamente combinados com diversos algoritmos de treinamento visando, ao final do teste, determinar a melhor combinação possível em termos de precisão, acurácia e tempo de execução.

1.1 OBJETIVOS

1.1.1 Objetivo geral

O objetivo geral deste projeto é testar combinações de algoritmos para classificação de anfíbios em um sistema composto de dois módulos, o módulo vetorizador e o módulo classificador.

1.1.2 Objetivos específicos

Para atingir o objetivo geral, este projeto propõe:

- Escolher um conjunto de dados adequado para avaliar a eficácia de vetorizadores e classificadores;
- Produzir uma quantidade razoável de funções vetorizadoras e classificadoras;
- Avaliar funções o desempenho de modelos de classificação de anfíbios por meio do desempenho de cada combinação possível entre funções vetorizadoras e classificadoras.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 REVISÃO DA LITERATURA

Múltiplos sistemas de monitoramento de anfíbios usando aprendizado de máquina como estratégia e gravações de chamados como material são documentados na literatura. (CHAO et al., 2019) apresenta o uso de uma rede neural acoplada com um vetorizador baseado em MFCCs, e compara melhores estratégias de computação baseadas em CPU e GPU.

Já (HUANG et al., 2009) obtém características pouco custosas e usa algoritmos relativamente rápidos, como kNN e SVM, para propor um sistema que funcione em uma plataforma online na qual cidadãos possam identificar anuros com sons gravados pelo celular.

Por sua vez, (GAN et al., 2021) escolhe o SVM, um classificador relativamente comum, mas decide por usar um método de vetorização baseado em espectrograma desenvolvido especialmente para a pesquisa, uma vez que ela tinha por objetivo reconhecer um gênero extremamente especializado e endêmico de sapos na Austrália cujas chamadas são muito semelhantes.

2.2 ETAPAS PARA A CONSTRUÇÃO DE UM SISTEMA

A maioria das pesquisas no ramo é baseada em cinco etapas principais: a coleta, a rotulagem, a vetorização, o treinamento e a aplicação. A coleta, a rotulagem e a vetorização são procedimentos fundamentais para a realização do treinamento, e a coleta, a vetorização e o treinamento, por sua vez, são procedimentos fundamentais para que a aplicação funcione.

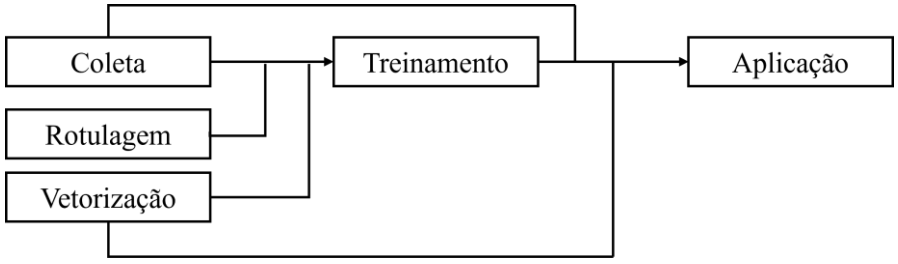

Figura 1 – etapas envolvidas. Fonte: elaborado pelo autor.

2.2.1 Coleta

Na coleta, deve-se obter um conjunto de sons de anuros de tamanho e variabilidade adequadas ao objetivo da pesquisa. A gravação dos áudios é geralmente feita com um módulo protegido dos elementos composto de um microfone programado para gravar de tempos em tempos ou mediante detecção, que transfere áudio para uma memória. Montagens mais complexas podem utilizar vários microfones, o que permite obter informações de localização e de intensidade (BLUMSTEIN et al., 2011).

Figura 2 – módulo de captação de áudio. Fonte: (CHAO et al., 2019).

De tempos em tempos, a memória é transferida para outros dispositivos. Opcionalmente, um algoritmo é utilizado para separar trechos de áudio relevantes e eliminar áudios espúrios, como motores e outros sons antropogênicos.

2.2.2 Rotulagem

Então, por meio do trabalho de profissionais especializados, cada arquivo ou trecho de áudio é rotulado com a sua respectiva espécie ou descartado caso não seja útil ao estudo. Essa etapa é sempre necessária para produzir um conjunto de treinamento e, por ser custosa, deve envolver o mínimo de amostras necessário para se obter uma representação completa dos anfíbios que habitam a região.

2.2.3 Vetorização

Arquivos de áudio são meras sequências de zeros e uns acompanhadas de metadados que permitem que um programa de computador as transforme em comandos de mais baixo nível que, em última instância, se transformarão em sinais elétricos em um

alto-falante. Por essa razão, eles não funcionam como boas entradas para algoritmos de aprendizado de máquina.

Para contornar esse problema, deve-se escolher um algoritmo de vetorização que será responsável por transformar cada arquivo de áudio em uma sequência numérica baseada em alguma propriedade acústica. Como essas sequências têm lastro em propriedades físicas teoricamente constantes dentro de uma mesma espécie e variáveis entre espécies diferentes, elas cumprem bem esse papel.

2.2.4 Treinamento e teste

Em seguida, deve-se escolher um algoritmo de aprendizado de máquina para classificação, treiná-lo e verificar seu desempenho através de métricas como acurácia, precisão e tempo de execução.

O processo começa dividindo o conjunto de vetores e seus rótulos em dois conjuntos aleatórios, chamados conjuntos de teste e treinamento, que não podem possuir elementos comuns, e possuem proporções pré-determinadas.

Então, o conjunto de treinamento é fornecido ao algoritmo de aprendizado de máquina supervisionado que usa uma técnica específica – a depender do algoritmo escolhido – para criar generalizações que permitam, ao obter dados sem rótulo, prever o rótulo correto.

Uma vez treinado, o algoritmo deve ser testado. Nessa etapa, são fornecidos apenas os vetores do conjunto de teste, e suas respostas em relação aos rótulos fornecidos comparados com os rótulos esperados são verificadas.

2.2.5 Aplicação

Uma vez que um algoritmo foi treinado e obteve um bom desempenho, ele pode ser finalmente utilizado para fins práticos. Nessa etapa, que corresponde ao ciclo de vida do projeto, áudios são captados pela etapa de coleta e metadados necessários (como data, hora e local) são incluídos em cada áudio.

Então, os áudios são enviados ao algoritmo de vetorização, e os vetores são enviados para o algoritmo de classificação, que produz rótulos, os quais são então combinados com os metadados, agregados e utilizados para diversos fins, como estudos em equilíbrio de ecossistemas ou mapeamento de populações nativas.

2.3 ARQUIVOS DE ÁUDIO

Arquivos de áudio são a base do projeto e, nesse caso, são recebidos no formato MPEG-4 Audio Layer (M4A). Nesse formato (WAGGONER, 2013), arquivos possuem um cabeçalho cuja função é identificar a extensão, seguido de metadados que podem incluir informações gerais sobre o arquivo, seguido de uma sequência de blocos correspondentes ao áudio comprimido utilizando geralmente Advanced Audio Coding (AAC).

Esse arquivo pode ser inserido em um decodificador, que será responsável por converter o áudio comprimido em um sinal discreto $S[n]$ – em outras palavras, uma sequência de números para áudios de um único canal, como o que este projeto usa – que representa a amplitude do sinal. Considerando N o total de amostras, podemos representar o sinal como:

$$S[n] = [s_0, s_1, \dots, s_{N-1}] \quad s_n \in \mathbb{R}$$

2.4 ALGORITMOS DE VETORIZAÇÃO

Algoritmos de vetorização são parte fundamental deste projeto e, por isso, é prudente discuti-los em aspectos matemáticos. Todos os modelos recebem um sinal $S[n]$, que possui um total de amostras N variável a depender do arquivo utilizado, e devolvem um vetor v.

2.4.1 Transformada curta de Fourier

A transformada curta de Fourier (STFT) permite converter um sinal de amplitudes no domínio do tempo para o domínio da frequência, e o algoritmo que leva seu nome usa esse resultado para obter uma média da amplitude em cada frequência ao longo do tempo (PIELEMEIER; WAKEFIELD; SIMONI, 1996).

A aplicação da STFT envolve a escolha de uma janela que terá comprimento M e tamanho de salto H, usualmente a janela de Hann, definida por:

$$w_{Hann}(n) = \frac{1}{2}\left(1 - \cos\left(\frac{2\pi n}{N}\right)\right)$$

Essa janela é aplicada ao áudio através de uma operação de convolução dentro da STFT, que é definida por:

$$S_1[f,t] = \sum_{n=0}^{M-1} S[n+tH] \times w_{Hann}(n) \times e^{-\frac{i2\pi fn}{M}}$$

O resultado é uma matriz $F[f,t]$ de dimensões $F \times T$, em que a dimensão F corresponde a valores discretos de frequência, e T corresponde a valores discretos de tempo. Na próxima etapa, obtemos o valor absoluto de cada valor complexo, ou seja:

$$S_2[f,t] = |S_1[f,t]|$$

Por fim, produzimos um vetor contendo a média ao longo do tempo de cada valor de frequência. Então:

$$v[f] = \frac{1}{T}\sum_{t=0}^{T-1} S_2[f,t]$$

2.4.2 MFCCs

Mel-frequency cepstral coefficients (MFCCs) é uma técnica que mapeia a potência do som, ou seja, a quantidade de energia transferida por segundo, em diversas frequências (A. MAJEED et al., 2015),

A aplicação do filtro envolve a escolha de uma janela que terá comprimento M e tamanho de salto H, nesse caso a janela de Hamming, definida por:

$$w_{Hamming}(n) = 0{,}54 - 0{,}46\cos\left(\frac{2\pi n}{N-1}\right)$$

Na próxima etapa, uma STFT é utilizada para converter a série temporal em uma série no domínio da frequência.

$$S_1[f,t] = \sum_{n=0}^{M-1} S[n+tH] \times w_{Hamming}(n) \times e^{-\frac{i2\pi fn}{M}}$$

Em seguida, a matriz é convertida para o espectro de potência – ou seja, o valor absoluto ao quadrado.

$$S_2[f,t] = \frac{|S_1[f,t]|^2}{N}$$

A escala Mel – cujo nome vem da palavra inglesa *melody* – é uma escala na qual a frequência, usualmente medida com uma escala linear em Hertz, é convertida em uma escala logarítmica em uma unidade arbitrária. Esse processo visa reproduzir a percepção humana de frequências.

Para converter $S_1[f,t]$ em escala de mel, deve-se multiplicar o vetor por um vetor $\mu[f_{mel},t]$ de U filtros triangulares que se intersectam e possuem bases cada vez maiores, e obter o logaritmo dos elementos do vetor resultante na base 10.

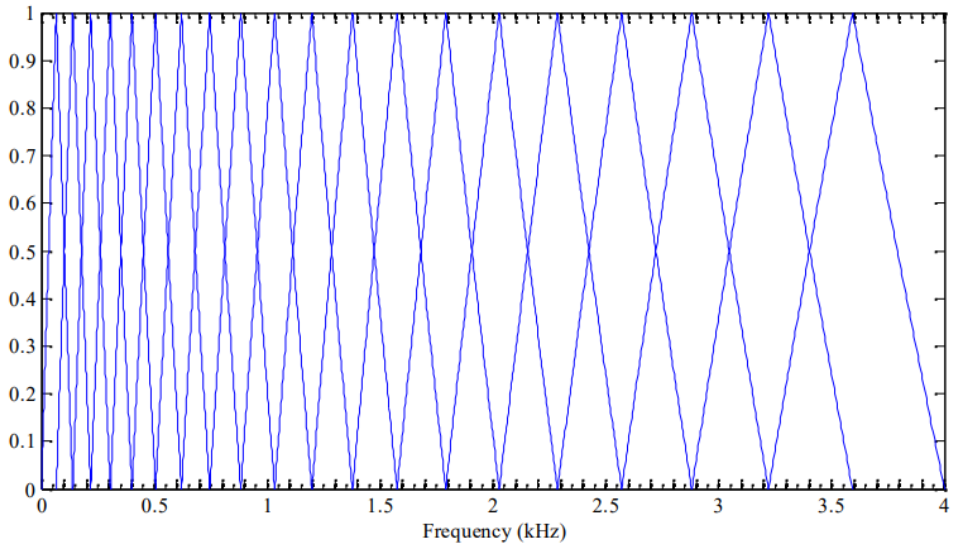

Figura 3 – filtro de mel. Fonte: (A. MAJEED et al., 2015)

Sua aplicação é matematicamente definida como:

$$S_2[f,t] = \log_{10}\left(\sum_{k=0}^{N-1} S_1[f,t] \times \mu[f_{mel},t]\right)$$

Finalmente, obtemos os coeficientes realizando a transformada discreta de cosseno.

$$S_3[f,t] = \sum_{u=1}^{U} S_2 \cos\left(\frac{\pi t}{U}\left(u - \frac{1}{2}\right)\right)$$

Depois, as primeiras e segundas derivadas de $S_3[f,t]$ em relação ao tempo são calculadas usando regressão linear.

$$S_3'[f,t] = \frac{\sum_{n-1}^{N} n \times (S_3[f,t+n] - S_3[f,t-n])}{2 \times \sum_{n-1}^{N} n^2}$$

$$S_3''[f,t] = \frac{\sum_{n-1}^{N} n \times (S_3'[f,t+n] - S_3'[f,t-n])}{2 \times \sum_{n-1}^{N} n^2}$$

Na última etapa, a média desses três vetores é calculada.

$$v[f] = \frac{1}{T}\sum_{t=0}^{T-1} \frac{S_3[f,t] + S_3'[f,t] + S_3''[f,t]}{3}$$

2.4.3 Rastreio de altura

O algoritmo de rastreio de altura extrai de um áudio as frequências nas quais há maior intensidade (MCFEE et al., 2015). Ele começa obtendo uma matriz de dimensões $F \times T$ definida como o módulo de uma STFT no sinal $S[n]$ usando a janela de Hann.

$$S_1[f,t] = \left| \sum_{n=0}^{M-1} S[n+tH] \times w_{Hann}(n) \times e^{-\frac{i2\pi fn}{M}} \right|$$

Em seguida, visando determinar os pontos de inflexão do sinal e suas magnitudes, dois cálculos são realizados. O primeiro obtém uma matriz de gradientes em relação ao eixo temporal.

$$S_G[f,t] = \frac{S[f,t+1] - S[f,t-1]}{2}$$

O segundo obtém uma matriz de pontos de inflexão, também em relação ao eixo temporal, calculando múltiplas parábolas que passam por três pontos conhecidos e sucessivos.

$$(x_n, y_n) = (t+n, S[t+n, f])$$

$$(x_{n+1}, y_{n+1}) = (t+n+1, S[t+n+1, f])$$

$$(x_{n+2}, y_{n+2}) = (t+n+2, S[t+n+1, f])$$

O cálculo das parábolas é feito por meio da determinação dos coeficientes a, b e c, obtidos na resolução de um sistema linear.

$$\begin{cases} ax_n^2 + bx_n + c = y_n \\ ax_{n+1}^2 + bx_{n+1} + c = y_{n+1} \\ ax_{n+2}^2 + bx_{n+2} + c = y_{n+2} \end{cases}$$

E, finalmente, a matriz de pontos inflexão pela fórmula conhecida.

$$S_V[f,t] = -\frac{b}{2a}$$

Então, ambos os vetores são interpolados para se obter as frequências e, em seguida, a média de amplitude de cada frequência é obtida.

$$S_2[f,t] = \frac{1}{2} \times S_G[f,t] \times S_V[f,t]$$

$$S_3[f] = \frac{1}{T}\sum_{t=0}^{T-1} S_2[f,t]$$

Finalmente, a saída é um vetor v que contém a média, desvio padrão, máximo, mínimo, mediana, primeiro quartil e terceiro quartil de S_3.

$$v = [\bar{S_3}, \sigma_{S_3}, \max S_3\ , \min S_3\ , \widetilde{S_3}, Q_1(S_3), Q_3(S_3)]$$

2.4.4 Croma

O algoritmo de croma extraí as intensidades em 12 classes de notas (MEINARD MÜLLER; KURTH; CLAUSEN, 2006). Ele começa calculando o quadrado do módulo de uma STFT ao sinal recebido usando a janela de Hann.

$$S_1[f,t] = \left| \sum_{n=0}^{M-1} S[n+tH] \times w_{Hann}(n) \times e^{-\frac{i2\pi fn}{M}} \right|^2$$

Então, a frequência base do som, f_{base}, é estimada realizando um rastreio de altura e obtendo o valor de maior intensidade. A partir dela, um banco de filtros $C[c,k]$ é produzido, em que $c \in [0, \ldots 11]$ e k corresponde a cada uma das notas.

Na próxima etapa, cada um dos filtros do banco C é aplicado no sinal.

$$S_2[f,t] = \sum_{c=0}^{11} C[c,k] \times S_1[f,t\ ]$$

Uma vez filtrado, o sinal é normalizado. O uso da constante ϵ, que simboliza um valor muito pequeno, é necessário para evitar a divisão por zero caso algum dos valores de S_2 seja igual à zero.

$$S_3[f,t] = \frac{S_2[f,t]}{\sum_{f'=0}^{11} S_2[f',t] + \epsilon}$$

O resultado, v, contém a média, o desvio padrão e máximo da matriz S_3.

$$v = [\overline{S_3}, \sigma_{S_3}, \max S_3]$$

2.4.5 Energia

A vetorização de energia utiliza uma janela para calcular o total de energia produzido pelo sinal de áudio ao longo de pequenos intervalos sucessivos e que se intercalam (PARIKH; SACHDEV, 2020).

Ela começa com a definição de uma janela, geralmente a quadrada.

$$w_{quadrada}(n) = \begin{cases} 1, & 0 \leq n < L \\ 0, & \text{caso contrário} \end{cases}$$

Então, ela é aplicada ao sinal $S[n]$, que produz um vetor de energia $S_1[n]$ seguindo a fórmula:

$$S_1[n] = \sqrt{\frac{1}{L}\sum_{n=0}^{L-1}\left(S[n+mH] \times w_{quadrada}[n]\right)^2}$$

Finalmente, $v[n]$ é obtido ajustando $S_1[n]$ para possuir R elementos, seja removendo os últimos elementos ou adicionando elementos nulos ao final até que ele possua R elementos.

2.5 ALGORITMOS DE CLASSIFICAÇÃO

A última parte do sistema é constituída de um algoritmo de classificação, que utiliza aprendizado de máquina supervisionado para determinar a classe de cada vetor recebido. Aqui, é considerado que o algoritmo na fase de treinamento recebe N instâncias, cada uma da forma $\mathrm{x} = (x_1, \ldots x_n)$, sendo n o número de atributos. Cada instância pode ser classificada em uma classe dentro de um vetor $\mathrm{y} = (y_1, \ldots y_k)$, sendo k o número de classes existentes.

Já no momento de classificação, o algoritmo recebe uma instância da forma $\mathrm{X} = (X_1, \ldots X_n)$, que deve ser classificada com uma classe $\hat{\mathrm{y}}$ dentro do vetor y.

2.5.1 Bayes ingênuo

O algoritmo de classificação Bayes ingênuo parte da premissa que cada uma das n variáveis presentes na instância são condicionalmente independentes – em outras palavras, não há intersecção na informação transmitida por cada variável (M NARASIMHA MURTY; V SUSHEELA DEVI, 2012).

Matematicamente, o algoritmo funciona criando k vetores $C_i(c_1, \dots, c_n)$, sendo k o total de classes, n o total de atributos, e c_i sendo a probabilidade de que aquela instância pertença à classe k segundo o atributo i.

Uma das formas mais comuns de se obter rótulo da classe devolvido pelo algoritmo, $\hat{y}$, é chamada de regra máximo a posteriori, dado pelo maior produto entre a fração de itens de classe C no conjunto de treinamento e o produto entre todos os elementos do vetor C_k.

$$\hat{y} = \max p(C_k) \prod_{i=1}^{n} p(X_i|C_k)$$

Neste projeto, utiliza-se o modelo de distribuição normal para as probabilidades, uma vez que essa é uma premissa bastante utilizada na estatística.

2.5.2 Máquina de vetores de suporte

A máquina de vetores de suporte (SVM) é um algoritmo cuja proposta envolve posicionar, durante o treinamento, as instâncias $(\mathrm{x}_1, y_1) \dots (\mathrm{x}_N, y_N)$ em um espaço vetorial, sendo $y \in 0, \dots k$ um valor inteiro para cada classe (NELLO CRISTIANINI; SHAWE-TAYLOR, 2004).

Então, na abordagem um-para-todos, utilizada neste projeto, um hiperplano dividindo cada par de duas classes é desenhado. Esse hiperplano é definido como:

$$\mathrm{w}^T \cdot \mathrm{x} + b = 0$$

Em que w é o vetor normal ao hiperplano e b é o viés, de forma que a equação acima indica que a operação de projeção de cada vetor x no vetor w somada do viés b resulta em zero.

Para problemas mais complexos, podemos fazer com que o hiperplano se torne não-linear no espaço de entrada, mas seja linear em um novo espaço obtido substituindo

o produto escalar por um núcleo, ou seja, por uma transformação linear entre dois espaços vetoriais.

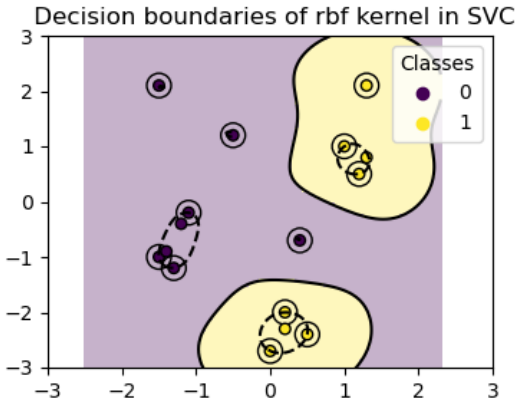

Figura 4 – exemplo de hiperplano em máquina de vetores de suporte. Fonte: ("SVM-Kernels", [s.d.]).

Esse núcleo, no caso utilizado nesse projeto, passa a ser definido como:

$$\mathrm{K}(\mathrm{x}_1, \mathrm{x}_2) = e^{-\gamma\|\mathrm{x}_1 - \mathrm{x}_2\|^2}$$

Uma vez definidos os hiperplanos, o processo de classificar novas instâncias envolve computar a função $f(\mathrm{x})$, em que α são pesos aprendidos durante o treinamento, para cada hiperplano traçado, de forma que cada cômputo produz um voto favorecendo uma das duas classes, e a classe vencedora $\hat{\mathrm{y}}$ é definida por voto majoritário.

$$f(\mathrm{x}) = \sum_{i=1}^{n} \alpha_i y_i K(\mathrm{x}_1 \mathrm{x}_2) + b$$

2.5.3 *k*-vizinhos mais próximos

O algoritmo k-vizinhos mais próximos (kNN) também envolve posicionar, durante o treinamento, as instâncias $(\mathrm{x}_1, y_1) \ldots (\mathrm{x}_N, y_N)$ em um espaço vetorial. Quando um novo ponto é recebido para classificação, deve-se computar a distância euclidiana desse ponto até cada um dos pontos classificados (GUO et al., 2003).

$$d(\mathrm{X}, x) = \sqrt{\sum_{i=0}^{n} (X_i - x_i)^2}$$

Então, as distâncias são ordenadas da menor para a maior, e os rótulos dos pontos k-ésimos primeiros pontos computam, cada um, um voto. A classe é definida pelo voto

majoritário, e o valor de k escolhido neste projeto, através de uma exploração intuitiva do desempenho do modelo, é de 7 vizinhos.

2.5.4 Análise de discriminante quadrático

Na análise de discriminante quadrático (QDA), é assumido que todas as classes possuem sua própria distribuição normal, isto é, cada classe está distribuída em um espaço vetorial de forma normal; e que todas as classes possuem sua própria matriz de covariância, isto é, a distribuição de cada classe possui uma forma diferente (THARWAT, 2016).

A partir do conjunto de treinamento, são calculados um vetor de média μ_k e uma matriz de covariância V_k para cada uma das k classes.

$$\mu_k = \frac{1}{n}\sum_{i-1}^{n} \mathrm{x}_i$$

$$V_k = \frac{1}{n-1}\sum_{i=1}^{n} (\mathrm{x}_i - \mu)(\mathrm{x}_i - \mu)^T$$

Então, quando recebemos um vetor X, calculamos

$$P(\mathrm{X}|y_k) = \frac{1}{2\pi^{1/2}|\mathrm{V}_k|^{1/2}} \times e^{\left(-\frac{1}{2}(\mathrm{X}-\mu_k)^T V^{-1}(X-\mu_k)\right)}$$

E, por fim, escolhemos a classe na qual a expressão $\ln P(\boldsymbol{X}|y_k)$ é maximizada. Neste projeto, utilizamos um padrão de regularização para minimizar *overfitting* de 0,1, escolhido através de consulta na literatura.

2.5.5 *Perceptron* multicamadas

Uma rede neural é composta de diversas camadas de neurônios individuais, também chamados de *perceptrons*. Cada *perceptron* funciona recebendo uma entrada na forma de um vetor v, e devolvendo um valor real z, calculado pela fórmula abaixo, em que w é um vetor de pesos (MURTAGH, 1991).

$$z = \mathrm{w}^T\mathrm{v} + b = b + \sum_{i=0}^{n} w_i v_i$$

Então, o neurônio aplica uma função de ativação em z, que determina o valor produzido como saída. Neste projeto, utilizamos a função unidade linear retificada (ReLU), que é dada pela fórmula:

$$\phi(z) = \max(0, z)$$

A rede neural utilizada neste projeto utiliza duas camadas intermediárias, com 100 e 50 neurônios respectivamente, escolhidos de forma intuitiva analisando o desempenho do modelo. Dessa forma, ao receber um novo vetor, a primeira camada calcula:

$$\mathrm{a}_1 = \max(0, w_1 X + \mathrm{b}_1)$$

Já a segunda camada calcula:

$$\mathrm{a}_2 = \max(0, w_2 a_1 + \mathrm{b}_2)$$

E, por fim, a terceira camada irá produzir um vetor que será convertido em um vetor de probabilidades para cada uma das k classes, da qual a com maior probabilidade será escolhida.

$$\mathrm{z} = w_3 a_2 + \mathrm{b}_3$$

$$\hat{y} = \mathrm{softmax}(\mathrm{z})_i = \frac{e^{z_i}}{\sum_{j=0}^{k} e^{z_j}}$$

Durante o treinamento, os valores dos pesos e dos vieses são ajustados para minimizar a função de perda, dada por:

$$\mathcal{L} = -\sum_{i=0}^{K} y_i (\log \hat{y}_k)$$

Em seguida, calcula-se o erro nas camadas 1 e 2, dado por:

$$\delta_2 = \hat{y} - y$$

$$\delta_1 = (\mathrm{w}_2 \delta_2) \circ \left(\phi'(z)\right)$$

Em que:

$$\phi'(z) = \begin{cases} 1 & z > 0 \\ 0 & z \leq 0 \end{cases}$$

Então, por meio do cômputo dos gradientes em relação ao peso e ao viés, podemos atualizá-los para cada camada considerando a taxa de aprendizado η da forma:

$$\mathrm{w}_1 \leftarrow \mathrm{w}_1 - \eta \frac{\partial \mathcal{L}}{\partial \mathrm{w}_1} = \mathrm{w}_1 - \eta \delta_1$$

$$\mathrm{b}_1 \leftarrow \mathrm{b}_1 - \eta \frac{\partial \mathcal{L}}{\partial \mathrm{b}_1} = \mathrm{b}_1 - \eta \delta_1 \mathrm{x}$$

$$\mathrm{w}_2 \leftarrow \mathrm{w}_2 - \eta \frac{\partial \mathcal{L}}{\partial \mathrm{w}_2} = \mathrm{w}_2 - \eta \delta_2$$

$$\mathrm{b}_2 \leftarrow \mathrm{b}_2 - \eta \frac{\partial \mathcal{L}}{\partial \mathrm{b}_2} = \mathrm{b}_2 - \eta \delta_2 \mathrm{a}_1$$

2.5.6 Floresta aleatória

Uma árvore de decisão é um diagrama de blocos que recebe um vetor e produz como resultado uma das possíveis classes. O treinamento do algoritmo da floresta aleatória começa treinando um número A de árvores de decisão com parâmetros aleatórios (BREIMAN, 2001).

Para cada árvore a ser treinada, o algoritmo primeiro escolhe um subconjunto aleatório de $N' < N$ vetores no conjunto de treinamento, bem como um subconjunto aleatório de atributos $n' < n$ que servirão para aquela árvore.

Então, esse subconjunto de vetores é usado para criar uma árvore. Ela é criada de forma recursiva, escolhendo um critério de divisão em um dos atributos que minimize a função da impureza de Gini, dada pela fórmula abaixo, em que p_i^2 denota a probabilidade de se obter duas instâncias sucessivas da mesma classe dentro do nó:

$$G(t) = 1 - \sum_{i=1}^{K} p_i^2$$

Uma vez que as A árvores estão construídas, um novo vetor X é classificado através de um voto majoritário entre elas. Neste projeto, utiliza-se 100 árvores, escolhido com base em consulta na literatura.

3 MATERIAIS E MÉTODOS

Neste capítulo, são discutidos os materiais e o método adotado para desenvolver o projeto para determinação da melhor combinação entre algoritmos de vetorização e de classificação. Para tanto, o capítulo está organizado da seguinte forma:

1. **Linguagem e bibliotecas**: são descritas todas as tecnologias utilizadas no desenvolvimento do projeto, como linguagens, estruturas de dados, bibliotecas de processamento de áudio e de aprendizado de máquina.
2. **Base de dados**: é descrita a origem dos dados utilizados e sua composição.
3. **Arquitetura do projeto**: será descrito todo o fluxo necessário para obter os resultados desejados.

Cada seção a seguir justifica as escolhas metodológicas utilizadas no projeto.

3.1 FERRAMENTAS

3.1.1 Linguagem e bibliotecas

Devido à vastidão de bibliotecas disponíveis e a possibilidade de programar em alto nível, a linguagem Python na versão 3.13.5 foi escolhida para realização do projeto.

Grande parte dos algoritmos auxiliares necessários para desenvolvimento do projeto já foram construídos em bibliotecas amplamente utilizadas na indústria e, por essa razão, podem ser utilizados a partir delas.

Dentre as bibliotecas de propósito geral, são utilizadas *os*[1], *zipfile*[2] e *urllib*[3] para, respectivamente, manipulação de arquivos dentro de um sistema operacional, extração de arquivos compactados e download de arquivos da internet. Todas essas bibliotecas são utilizadas na mesma versão que a linguagem Python, isto é, 3.13.5.

O processo de vetorização é feito primariamente com algoritmos da biblioteca *Librosa*[4] na versão 0.11.0, mas também com funções auxiliares das bibliotecas *NumPy*[5] na versão 2.3 e *SciPy*[6] na versão 1.16.0. Uma vez que os vetores são produzidos, a criação

[1] Disponível em https://docs.python.org/3/library/os.html.
[2] Disponível em https://docs.python.org/3/library/zipfile.html.
[3] Disponível em https://docs.python.org/3/library/urllib.html.
[4] Disponível em https://librosa.org/doc/latest/index.html.
[5] Disponível em https://numpy.org/.
[6] Disponível em https://scipy.org/.

de um conjunto de vetores para aprendizado é auxiliada pelas bibliotecas Pandas, para organização dos dados, e Re, para processamento de texto.

Por fim, o processo de classificação é feito unicamente utilizando os algoritmos da biblioteca *Scikit-learn*[7] na versão 1.7.1, que incluem tanto rotinas para treinamento e aplicação do modelo quanto para sua avaliação.

3.1.2 Base de dados

A base de dados escolhida para o experimento contém 1536 sons de anuros pertencentes a 26 espécies diferentes (AKBAL et al., 2023) disponível para uso no Kaggle[8]. Cada som está em um arquivo no formato M4A, e possui um nome no qual consta a espécie do anuro (como um número entre 0 e 25), seguido do código do som entre parênteses (como um número entre 0 e 1535).

Tabela 1 – códigos associados aos nomes científicos e vulgares de cada anfíbio, bem como o total de arquivos e a duração total. Fonte: (AKBAL et al., 2023), adaptado pelo autor.

ID	Nome científico	Nome vulgares	Número de sinais	Duração total em segundos
0	*Acris gryllus*	Rã-grilo-do-sul	70	177,36
1	*Agalychnis callidryas*	Rã-de-olhos-vermelhos	85	244,26
2	*Colostethus talamancae*	Rã-de-foguete	52	154,53
3	*Alytes Cisternasii*	Sapo-parteiro-ibérico	50	143,95
4	*Alytes muletensis*	Sapo-parteiro-de-maiorca	30	74,43
5	*Ameerega cainarachi*	Sapo-venenoso-Cainarachi	75	188,15
6	*Anaxyrus cognatus*	Sapo-das-grandes-planícies	53	141,11
7	*Anaxyrus quercicus*	Sapo-do-carvalho	50	115,58
8	*Centrolene savage*	Sapo-de-vidro	50	146,71
9	*Dendrobates leucomelas*	Sapo-venenoso-da-faixa-amarela	50	130,40

[7] Disponível em https://scikit-learn.org/stable/.
[8] Disponível em https://www.kaggle.com/datasets/mehmetbayin/anuran-sound-frogs-or-toads-dataset.

10	*Dendrobates tinctorius azureus*	Sapo-venenoso-azul	40	110,24
11	*Eleutherodactylus pallidus*	Sapo-pálido	50	134,42
12	*Eleutherodactylus unistrigatus*	Sapo-ladrão-listrado	42	107,86
13	*Gastrotheca riobambae*	Perereca-marsupial-andina	55	159,84
14	*Hyla squirella*	Perereca-esquilo	70	171,96
15	*Hylarana laterimaculata*	Sapo-do-pântano-de-manchas-laterais	66	203,28
16	*Kaloula pulchra*	Rã-touro-listrada	81	204,47
17	*Microhyla butleri*	Sapo-de-coro-pintado	76	180,11
18	*Microhyla heymonsi*	Sapo-de-coro-de-lados-escuros	50	134,39
19	*Microhyla ornata*	Sapo-de-boca-estreita-ornamentado	72	179,79
20	*Oophaga granulifera*	Rã-venenosa-granulada	73	190,41
21	*Oophaga pumili*	Rã-morango	70	189,80
22	*Rhinella Marina*	Sapo-cururu	70	205,41
23	*Scaphiopus couchii*	Sapo-de-pá-de-Couch	50	133,78
24	*Smilisca baudinii*	Rã-de-árvore-mexicana-comum	51	144,38
25	*Spea multiplicata*	Sapo-de-pá-do-Novo-México	55	136,29

3.1.3 Arquitetura do projeto

O fluxo do projeto é organizado em diversas etapas. A primeira se resume a importar todas as bibliotecas necessárias, baixar uma chave de acesso ao Kaggle e baixar a base de dados utilizando a chave de acesso. Em seguida, o projeto é realizado no Google Colab.[9]

[9]Disponível em https://colab.research.google.com/drive/1hh0-vuH1tQlzhoTaCDwcQ3tSRNuoswKx.

Na segunda etapa, são construídos cinco *dataframes*, cada um resultado da aplicação de um dos cinco algoritmos de vetorização. Cada *dataframe* servirá de base para executar os cinco algoritmos de aprendizado de máquina.

Na terceira e última etapa, itera-se cada *dataframe* com cada função de classificação, totalizando trinta possíveis combinações. Cada execução produz um valor de precisão, acurácia e revocação, os quais são exibidos em uma tabela.

O pseudocódigo a seguir mostra, em alto nível e sintaxe próxima do linguajar natural, como o fluxo de execução ocorre.

Pseudocódigo 1: execução principal do projeto

```
# Primeira etapa, preparação preliminar.
Importar bibliotecas.
Baixar chave do Kaggle a partir de endereço web.
Baixar conjunto de dados a partir da chave do Kaggle e do nome do conjunto.
Extrair arquivos de áudio do conjunto de dados em uma etapa.

# Segunda etapa, criação dos dataframes.
Para cada algoritmo de vetorização:
    Para cada arquivo de áudio:
        Calcular o vetor equivalente ao arquivo usando o algoritmo de vetorização.
        Inserir o arquivo em um dataframe.
    Salvar o dataframe.

# Terceira etapa, aplicação dos algoritmos de classificação e avaliação.
Para cada algoritmo de classificação:
    Para cada dataframe:
        Construir um modelo de classificação.
        Separar o dataframe em 30% para teste e 70% para treinamento.
        Treinar o modelo de classificação.
        Testar o modelo utilizando validação cruzada com cinco repartições.
        Calcular precisão.
        Calcular acurácia.
        Calcular revocação.
```

4 ANÁLISE EXPERIMENTAL E DISCUSSÃO

Nessa seção, descreve-se o experimento realizado para investigar o desempenho de cada uma das 30 possíveis combinações. A partir da análise realizada, busca-se responder questões como:

- Dentre as combinações, qual possui a melhor acurácia, a melhor precisão e a melhor revocação?
- Os valores de acurácia, precisão e revocação permitem apontar que combinações diferentes são melhores para propósitos diferentes?

4.1 CONFIGURAÇÃO EXPERIMENTAL

Para realização do experimento, cada algoritmo de vetorização e cada modelo de aprendizado foi preparado utilizando os parâmetros descritos anteriormente, e uma fração de 30% do conjunto foi utilizada para teste, com 70% utilizada para treinamento.

Para a avaliação das combinações, foi realizada uma validação cruzada com 5 repartições dentro do conjunto de teste, ou seja, o conjunto de teste (30% do total) foi dividido em 5 repartições aleatórias, cada uma foi utilizada para um teste, e a média das métricas foi calculada.

4.2 MÉTRICAS DE AVALIAÇÃO

Aqui, três métricas relativas ao sucesso do algoritmo (RAINIO; TEUHO; KLÉN, 2024) são relevantes.

4.2.1 Matriz de confusão

Em uma tarefa de classificação com k classes, uma matriz de confusão é uma matriz $k \times k$ na qual o elemento n_{ij} representa o número de instâncias da i-ésima classe classificadas como instâncias da j-ésima classe. Dessa matriz, quatro variáveis são extraídas.

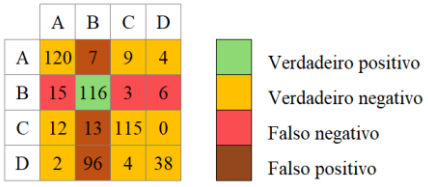

Figura 5 – Matriz de confusão com quatro classes a partir da classe B. Fonte: elaborado pelo autor, adaptado de (RAINIO; TEUHO; KLÉN, 2024).

Um verdadeiro positivo é definido como toda instância pertencente à classe i classificada como pertencente à classe i. Dessa forma:

$$TP_i = n_{ii} \quad i = 1, \dots, k$$

Um verdadeiro negativo é definido como toda instância não pertencente à classe i classificada como não pertencente à classe i. Dessa forma:

$$TN_i = \sum_{j \neq i} \sum_{h \neq i} n_{jh} \quad i, j, h = 1, \dots, k$$

Um falso positivo é definido como toda instância não pertencente à classe i classificada como pertencente à classe i. Dessa forma:

$$FP_i = \sum_{j \neq i} n_{ij} \quad i, j = 1, \dots, k$$

Um falso negativo é definido como toda instância pertencente à classe i classificada como não pertencente à classe i. Dessa forma:

$$FN_i = \sum_{j \neq i} n_{ij} \quad i, j = 1, \dots, k$$

Após serem calculadas para cada classe, essas variáveis são definidas para o algoritmo. Isso pode ser feito utilizando média simples, também chamada de *macro averaging*, ou utilizando média ponderada pelo número de instâncias na classe, também chamada de *micro averaging*. Neste projeto, utilizou-se *micro averaging*.

4.2.2 Acurácia

A acurácia representa a relação entre o total de previsões corretas e o total de previsões realizadas. Dessa forma:

$$\text{Acc} = \frac{TP}{TP + TN + FP + FN}$$

A acurácia é uma métrica simples de ser interpretada, mas que desconsidera detalhes importantes como o balanceamento das classes, em outras palavras, se há baixa taxa de sucesso em classes específicas.

4.2.3 Precisão

A precisão é calculada através de média simples ou ponderada das relações entre o total de previsões atribuídas corretamente à uma classe e o total de previsões, corretas ou não, atribuídas à essa classe. Dessa forma:

$$\text{Pre} = \frac{TP}{TP + FP}$$

A precisão é um bom indicador da capacidade do modelo em identificar cada classe corretamente.

4.2.4 Revocação

A revocação é calculada através de média simples ou ponderada das entre o total de previsões atribuídas corretamente à uma classe e o total de instâncias pertencentes àquela classe. Dessa forma:

$$\text{Rev} = \frac{TP}{TP + FN}$$

A revocação é uma boa indicadora da capacidade do modelo em identificar a ocorrência de cada classe.

4.3 RESULTADOS

Os resultados dos experimentos estão nas tabelas abaixo. Cada uma das tabelas representa uma métrica para as combinações de algoritmos de vetorização (eixo horizontal) com algoritmos de classificação (eixo vertical).

Tabela 2 – resultado da métrica de precisão.

	MFCC	**Altura**	**Croma**	**Energia**	**STFT**
QDA	0,93	0,24	0,83	0,25	0,91
SVM	0,92	0,41	0,83	0,33	0,89
***k*NN**	0,84	0,87	0,80	0,29	0,93
Perceptron	0,93	0,59	0,85	0,38	0,98
Bayes	0,87	0,27	0,51	0,24	0,84
Floresta	0,98	0,70	0,85	0,55	0,98

Tabela 3 – resultado da métrica de acurácia.

	MFCC	**Altura**	**Croma**	**Energia**	**STFT**
QDA	0,93	0,34	0,82	0,22	0,82
SVM	0,91	0,45	0,82	0,30	0,82
***k*NN**	0,82	0,57	0,79	0,30	0,92
Perceptron	0,93	0,59	0,84	0,39	0,98
Bayes	0,84	0,34	0,37	0,27	0,77
Floresta	0,98	0,69	0,85	0,54	0,98

Tabela 4 – resultado da métrica de revocação.

	MFCC	**Altura**	**Croma**	**Energia**	**STFT**
QDA	0,92	0,31	0,80	0,18	0,80
SVM	0,90	0,42	0,81	0,27	0,81
***k*NN**	0,81	0,55	0,79	0,28	0,92
Perceptron	0,93	0,57	0,84	0,37	0,98
Bayes	0,84	0,32	0,37	0,27	0,77
Floresta	0,98	0,68	0,84	0,53	0,98

4.4 DISCUSSÃO DOS RESULTADOS

Verifica-se que os resultados de precisão, acurácia e revocação para uma mesma combinação de vetorizador e classificador são suficientemente semelhantes entre si, já que a variação entre eles, na maioria dos casos, fica restrita à segunda casa decimal. Em outras palavras, os modelos possuem um desempenho semelhante em relação aos erros cometidos ao identificar falsos positivos e falsos negativos.

Além disso, pode-se tornar a compreensão das tabelas acima mais fáceis ao atribuir uma escala de desempenho às combinações. Consideramos a menor métrica entre precisão, acurácia e revocação, e aplicamos a escala:

- Ruim, caso o valor seja menor que 0,50 (vermelho)
- Regular, caso o valor seja maior ou igual à 0,50, mas menor que 0,80 (amarelo);
- Bom, caso o valor seja maior ou igual à 0,80, mas menor que 0,90 (verde);
- Muito bom, caso o valor seja maior ou igual à 0,90, mas menor que 0,95 (azul);
- Excepcional, caso o valor seja maior que 0,95 (roxo).

Os resultados dessa escala estão na tabela abaixo.

Tabela 5 – aplicação da escala nos resultados obtidos

	MFCC	**Altura**	**Croma**	**Energia**	**STFT**
QDA	0,92	0,24	0,80	0,18	0,80
SVM	0,90	0,41	0,81	0,27	0,81
***k*NN**	0,81	0,55	0,79	0,28	0,92
Perceptron	0,93	0,57	0,84	0,37	0,98
Bayes	0,84	0,27	0,37	0,24	0,77
Floresta	0,98	0,68	0,84	0,53	0,98

Dessa forma, MFCC e STFT são algoritmos de vetorização que apresentam bons resultados com qualquer modelo de aprendizado de máquina. Ambos trabalham com faixas de frequências, sendo que a STFT é o algoritmo que envolve a menor quantidade de etapas de processamento, o que indica que os melhores atributos a se analisar são espectrais. Além disso, para *perceptron* multicamadas e floresta aleatória, os melhores resultados, com taxas de até 0,98, são obtidos.

Além disso, a métrica de croma apresentou um desempenho intermediário entre todos classificadores. Já as métricas de altura e energia apresentaram, em geral, um desempenho ruim.

O desempenho apresenta grande variação horizontal (ou seja, no eixo relativo aos algoritmos de vetorização) e pouca variação vertical (ou seja, no eixo relativo aos algoritmos de classificação), o que indica que altura, croma e energia não são bons indicadores para classificação de anuros – em oposição à MFCC e STFT – e que os classificadores apresentam um bom desempenho geral quando um bom algoritmo de vetorização é usado.

5 CONCLUSÃO

Neste trabalho, foram exploradas 30 estratégias para a classificação de anuros através do som, sendo elas produto da combinação entre cinco algoritmos de vetorização (MFCC, STFT, altura, croma e energia) e seis algoritmos de classificação (análise de discriminante quadrático, máquina de vetores de suporte, k-vizinhos mais próximos, *perceptron* multicamadas e Bayes ingênuo.

Os resultados indicam que MFCC e STFT apresentam bons resultados com qualquer algoritmo de classificação, mas que resultados excepcionais são obtidos combinando MFCC com floresta aleatória, e STFT com floresta aleatória e *perceptron* multicamadas.

5.1 LIMITAÇÕES

Apesar dos avanços obtidos, as conclusões apresentam algumas limitações que devem ser consideradas:

- **Especificidades dos dados**: o estudo englobou gravações de sapos, rãs e pererecas, o que implica em maior distância evolutiva e, portanto, uma tendência em diferenças no aparelho vocal e, consequentemente, no som produzido. Um estudo com indivíduos próximos em caráter evolutivo pode indicar que outra combinação é mais adequada.
- **Fluxo de dados**: o estudo foi feito em um conjunto de dados no qual o volume não era um problema. Em aplicações de grande escala, em que é necessário processar grande volume de dados em um tempo curto, pode-se optar por uma pior precisão caso essa desvantagem seja aceitável frente a um melhor tempo de execução quando um algoritmo é paralelizado.

5.2 TRABALHOS FUTUROS

Para superar as limitações apontadas na seção anterior, sugerem-se as seguintes áreas de pesquisa:

- **Testes com dados mais refinados**: a base de dados usada no estudo inclui espécies do mundo todo, e não anuros endêmicos de uma região ou evolutivamente próximos. Mediante a obtenção desse tipo de base de dados, pode-se determinar qual combinação é melhor em cada situação.

- **Testes com alto fluxo de dados**: ao impor restrições de tempo, pode-se explorar como cada algoritmo em ambas as categorias pode ser acelerado com paralelização, e checar se uma pequena perda de desempenho compensa um ganho em velocidade de execução.
- **Testes com sistemas embarcados**: visando minimizar a necessidade de extração periódica dos arquivos, pode-se estudar combinações de sistemas embarcados para realização da etapa de vetorização e classificação no local, com transmissão remota de dados.
- **Elaboração das demais etapas**: a pesquisa pode ser expandida para incluir, também, comparativo das melhoras técnicas a serem utilizadas nas etapas não abordadas neste estudo.

5.3 CONSIDERAÇÕES FINAIS

Transformada de Fourier de curto termo e cepstrum de frequência Mel são bons algoritmos de vetorização quando combinados com todos os algoritmos de classificação testados, mas apresentam resultados excepcionais com *perceptron* multicamadas e floresta aleatória.

Esses avanços apontam para um caminho promissor de identificação de sons de anuros o que, por sua vez, aponta para inúmeras aplicações de baixo custo para monitoramento ambiental.

REFERÊNCIAS

A. MAJEED, S. et al. Mel frequency cepstral coefficients (Mfcc) feature extraction enhancement in the application of speech recognition: A comparison study. **Journal of theoretical and applied information technology**, v. 79, p. 38–56, 10 set. 2015. Disponível em: https://www.semanticscholar.org/paper/Mel-frequency-cepstral-coefficients-(Mfcc)-feature-Majeed-Husain/0373c466d7b19d030e6b311dec184d53c8c4c62a.

ANDERSEN, D.; BORZÉE, A.; JANG, Y. Predicting global climatic suitability for the four most invasive anuran species using ecological niche factor analysis. **Global Ecology and Conservation**, v. 25, p. e01433, jan. 2021. Disponível em: https://www.sciencedirect.com/science/article/pii/S2351989420309756.

AKBAL, E. et al. Explainable automated anuran sound classification using improved one-dimensional local binary pattern and Tunable Q Wavelet Transform techniques. **Expert Systems with Applications**, v. 225, p. 120089, 13 abr. 2023. Disponível em: https://www.sciencedirect.com/science/article/pii/S0957417423005912.

BLUMSTEIN, D. T. et al. Acoustic monitoring in terrestrial environments using microphone arrays: applications, technological considerations and prospectus. **Journal of Applied Ecology**, v. 48, n. 3, p. 758–767, 6 abr. 2011. Disponível em: https://besjournals.onlinelibrary.wiley.com/doi/10.1111/j.1365-2664.2011.01993.x.

BREIMAN, L. Random Forests. **Machine Learning**, v. 45, n. 1, p. 5–32, out. 2001. Disponível em: https://link.springer.com/article/10.1023/A:1010933404324.

CHAO, K.-W. et al. Using Machine Learning Method to Identify for Frog Classification. **2019 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE)**, p. 168–171, 1 out. 2019. Disponível em: https://ieeexplore.ieee.org/document/8942750.

DEPARTMENT OF THE ENVIRONMENT, WATER, HERITAGE AND THE ARTS. **The cane toad (Bufo marinus) - fact sheet**. Disponível em: https://www.dcceew.gov.au/environment/invasive-species/publications/factsheet-cane-toad-bufo-marinus

GAN, H. et al. A novel frog chorusing recognition method with acoustic indices and machine learning. **Future Generation Computer Systems**, v. 125, p. 485–495, dez. 2021.

GUO, G. et al. KNN Model-Based Approach in Classification. **On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE**, v. 2888, p. 986–996, 2003. Disponível em: https://link.springer.com/chapter/10.1007/978-3-540-39964-3_62.

HUANG, C.-J. et al. Frog classification using machine learning techniques. **Expert Systems with Applications**, v. 36, n. 2, p. 3737–3743, mar. 2009. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0957417408001504.

JOSEPH MUSONDA KABANZE et al. Anthropogenic effects of habitat modification on anuran species diversity in a swamp forest area, Kenya. **African Journal of Ecology**, v. 62, n. 1, 19 dez. 2023. Disponível em: https://onlinelibrary.wiley.com/doi/10.1111/aje.13245.

M NARASIMHA MURTY; V SUSHEELA DEVI. **Pattern recognition: an algorithmic approach**. London: Springer, 2012.

MCFEE, B. et al. librosa: Audio and Music Signal Analysis in Python. **Proceedings of the 14th Python in Science Conference**, 2015. Disponível em: https://www.semanticscholar.org/paper/librosa%3A-Audio-and-Music-Signal-Analysis-in-Python-McFee-Raffel/e5c114afc8c4d4e10ae068ba8e3387cc13e17a6e.

MEINARD MÜLLER; KURTH, F.; CLAUSEN, M. Croma-based statistical audio features for audio matching. 11 out. 2006. Disponível em: https://ieeexplore.ieee.org/document/1540223.

MURTAGH, F. Multilayer perceptrons for classification and regression. **Neurocomputing**, v. 2, n. 5-6, p. 183–197, jul. 1991. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/0925231291900235.

NELLO CRISTIANINI; SHAWE-TAYLOR, J. **Support vector machines and other kernel-based learning methods.** Cambridge (Inglaterra): Cambridge, 2004.

PARIKH, D.; SACHDEV, S. Improving the efficiency of spectral features extraction by structuring the audio files. **arXiv (Cornell University)**, p. 1–5, 11 set. 2020. Disponível em: https://arxiv.org/pdf/2010.03136.

PIELEMEIER, W. J.; WAKEFIELD, G. H.; SIMONI, M. H. Time-frequency analysis of musical signals. **Proceedings of the IEEE**, v. 84, n. 9, p. 1216–1230, 1996. Disponível em: https://ieeexplore.ieee.org/document/535242.

RAINIO, O.; TEUHO, J.; KLÉN, R. Evaluation metrics and statistical tests for machine learning. **Scientific Reports**, v. 14, n. 1, p. 1–14, 13 mar. 2024. Disponível em https://www.nature.com/articles/s41598-024-56706-x.

SAHIDULLAH, MD.; SAHA, G. Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition. **Speech Communication**, v. 54, n. 4, p. 543–565, maio 2012. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0167639311001622.

STARNBERGER, I.; PREININGER, D.; HÖDL, W. The anuran vocal sac: a tool for multimodal signalling. **Animal Behaviour**, v. 97, p. 281–288, nov. 2014. Disponível em https://pmc.ncbi.nlm.nih.gov/articles/PMC4222773/.

SVM-Kernels. Disponível em: https://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html#sphx-glr-auto-examples-svm-plot-svm-kernels-py.

THARWAT, A. Linear vs. quadratic discriminant analysis classifier: a tutorial. **International Journal of Applied Pattern Recognition**, v. 3, n. 2, p. 145, 2016. Disponível em: https://www.inderscienceonline.com/doi/abs/10.1504/IJAPR.2016.079050.

WAGGONER, B. **Compression for Great Video and Audio**. [s.l.] Taylor & Francis, 2013.