



INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA (ISEL)

DEPARTAMENTO DE ENGENHARIA ELETRÓNICA E DE  
TELECOMUNICAÇÕES E COMPUTADORES (DEETC)

---

LEIM

LICENCIATURA EM ENGENHARIA INFORMÁTICA E MULTIMÉDIA  
UNIDADE CURRICULAR DE PROJETO

---

## Anotação de Eventos Sonoros em Vídeo

Carina Fernandes (45118)

---

*Orientadores*

*Professor Doutor*      Joel Paulo

*Professor Doutor*      Paulo Trigo

*Professor*                Paulo Vieira

---

*Setembro, 2022*



# Resumo

O desempenho de um atleta, numa determinada modalidade desportiva, melhora quando é acompanhado de uma perspetiva externa no decurso da sua atividade desportiva. O atleta pode ser monitorizado por um treinador para atingir melhores resultados. Como complemento, é possível registar em vídeo e analisar posteriormente o seu desempenho. Neste sentido, é vantajoso o desenvolvimento de uma ferramenta capaz de realizar essa análise externa.

A ferramenta desenvolvida neste projeto permite extrair e reconhecer batidas de bola em raquete com base no vídeo da atividade desportiva do atleta (em treinos ou competições de padel /ténis).

Trata-se de um problema de classificação binário em que o sistema deve fazer a distinção entre batidas de bola e ruído. Estes eventos têm uma duração típica pré-determinada e a distinção entre eles foi feita com base nas características (*Onset Detect*, *Root Mean Square* e *Spectral Flux*) identificadas no áudio através de técnicas de aprendizagem automática.

O *dataset* utilizado no processo de aprendizagem foi construído de raiz e o desequilíbrio verificado neste conjunto de dados foi abordado através de um processo de sub-amostragem.

Entre os vários métodos de aprendizagem automática analisados, escolheu-se a rede neuronal do tipo perceptrão multi-camada (*Multi Layer Perceptron* - MLP) para realizar a discriminação entre os tipos de eventos mencionados.

No final, através de uma aplicação *web*, são visualizados os resultados repetitantes aos instantes em que ocorrem batidas de bola, dando uma perspetiva mais abrangente e objetiva do desempenho do atleta. A ferramenta identifica corretamente cerca de 85% dos eventos em análise.

Os ensaios são realizados no Laboratório de Áudio e Acústica do ISEL, LAA.

**Palavras-chave:** algoritmos de software de deteção do som, algoritmos de software de inteligência artificial.



# Abstract

In a given sport, the athlete's performance improves when he is supervised from an external perspective. In this scenario, the athlete can be supervised by a coach in order to achieve better results, or as a complement, it is possible to video record and analyze his performance afterwards. Thus, it is advantageous to develop a tool capable of performing this external analysis.

This is a binary classification problem in which the system must distinguish between ball hits and noise. These events have a predetermined typical duration and this distinction was made based on the features (*Onset Detect*, *Root Mean Square* and *Spectral Flux*) identified in the audio through machine learning techniques.

The *dataset* used in the learning process was built from scratch and the imbalance found in this dataset was addressed through an oversampling process.

Among the various machine learning methods analyzed, the multi-layer perceptron (MLP) was chosen to perform the discrimination between the mentioned event types.

At the end, through a *web* application, the results relative to the instants in which ball hits occur are visualized, giving a more comprehensive and objective perspective of the athlete's performance. The tool correctly identifies about 85% of the events under analysis, with some adjustments needed to improve the recognition process.

The tests are carried out at ISEL's Audio and Acoustics Laboratory, LAA.

**Keywords:** sound detection software algorithms, artificial intelligence software algorithms.



# Agradecimentos

À minha família, especialmente à minha mãe e os meus irmãos, pelo apoio e incentivo concedidos durante a realização deste projeto.

Aos meus orientadores, Joel Paulo, Paulo Trigo e Paulo Vieira, sem os quais a realização deste projeto não seria possível.

A mim mesma pela coragem de não desistir apesar dos desafios inerentes ao projeto.





# Índice

Resumo	i
Abstract	iii
Agradecimentos	v
Índice	ix
Lista de Tabelas	xi
Lista de Figuras	xiii
<b>1 Introdução</b>	<b>1</b>
1.1 Motivação . . . . .	1
1.2 Principais Contributos . . . . .	2
1.3 Estrutura do Relatório . . . . .	2
<b>2 Fundamentos</b>	<b>5</b>
2.1 Conceitos de Processamento de Sinal . . . . .	5
2.1.1 Amostragem . . . . .	5
2.1.2 Transformada de Fourier . . . . .	6
2.1.3 Características extraídas do áudio . . . . .	6
2.2 Conceitos de Aprendizagem Automática . . . . .	7
2.2.1 <i>Dataset</i> . . . . .	8
2.2.2 Desiquilíbrio no <i>Dataset</i> . . . . .	8
2.2.3 Redes Neurais . . . . .	9
2.2.4 Hiper-parâmetros . . . . .	10
2.2.5 Sobre-aprendizagem . . . . .	10

2.2.6	Validação Cruzada . . . . .	11
<b>3</b>	<b>Trabalho Relacionado</b>	<b>15</b>
3.1	Anotação de eventos . . . . .	15
3.2	Extração de Características e Reconhecimento de Padrões . . .	16
<b>4</b>	<b>Modelo Proposto</b>	<b>17</b>
4.1	Descrição Geral do Sistema . . . . .	17
4.2	Requisitos . . . . .	17
4.3	Abordagem . . . . .	19
4.3.1	Construção do <i>Dataset</i> . . . . .	19
4.3.2	Construção do Classificador . . . . .	22
4.3.3	Desenvolvimento da Aplicação <i>Web</i> . . . . .	22
<b>5</b>	<b>Implementação do Modelo</b>	<b>25</b>
5.1	Construção do <i>Dataset</i> . . . . .	25
5.2	Construção do Classificador . . . . .	26
5.3	Desenvolvimento da Aplicação <i>web</i> . . . . .	28
<b>6</b>	<b>Validação e Testes</b>	<b>31</b>
6.1	Definição dos Hiper-parâmetros . . . . .	31
6.2	Classificação de Vídeos Novos . . . . .	34
6.3	Anotação e Melhoria do <i>Dataset</i> . . . . .	35
<b>7</b>	<b>Conclusões e Trabalho Futuro</b>	<b>37</b>
7.1	Conclusões . . . . .	37
7.2	Trabalho Futuro . . . . .	38
<b>A</b>	<b>Um Detalhe Adicional</b>	<b>39</b>
<b>B</b>	<b>Outro Detalhe Adicional</b>	<b>41</b>
	<b>Bibliografia</b>	<b>43</b>

# Lista de Tabelas

4.1	Requisitos funcionais do sistema em desenvolvimento para a construção do <i>dataset</i> . . . . .	18
4.2	Requisitos funcionais respeitantes ao processo de classificação. . . . .	18
4.3	Requisitos não funcionais do sistema. . . . .	18
4.4	Valores de duração (em segundos), amostras delizadas e N a considerar, para um valor de <i>samplingRate</i> igual a 44100Hz. . . . .	20
6.1	Resultados obtidos para os vários hiper-parâmetros da rede neuronal e <i>datasets</i> diferentes. . . . .	33
6.2	Resultados da classificação sobre vídeos com características diferentes. . . . .	34



# Lista de Figuras

2.1	Características (cima e meio) e áudio (baixo) correspondente. .	7
2.2	Representação típica de uma rede neuronal. . . . .	9
4.1	Sistema de anotação de eventos desejado. . . . .	17
4.2	Matriz de características. . . . .	20
4.3	Vetor de classes. . . . .	21
4.4	Matriz de características. . . . .	22
4.5	A da aplicação <i>web</i> . . . . .	23
4.6	<i>Design</i> desejado da página <i>web</i> 1. . . . .	23
4.7	<i>Design</i> desejado da página <i>web</i> 2. . . . .	24
5.1	Processo detalhado de construção do <i>dataset</i> . . . . .	26
5.2	Processo de construção do classificador. . . . .	27
5.3	Resultados (independentes das classes) da aplicação de vários algoritmos aos dados. . . . .	27
5.4	Processo de desenvolvimento da aplicação <i>web</i> . . . . .	29
5.5	Estrutura do projeto correspondente à aplicação <i>web</i> no IDE Netbeans. . . . .	29
6.1	Matrizes de confusão resultantes da classificação dos vídeos A (esquerda), B (meio) e C (direita). . . . .	34
6.2	Interface da página 2 da aplicação <i>web</i> . . . . .	35
B.1	Descrição pormenorizada do campo de padel. [Tennisnerd, 2015]	41
B.2	Resultados da aplicação de vários algoritmos aos dados para a classe dos positivos. . . . .	42
B.3	Resultados da aplicação de vários algoritmos aos dados para a classe dos negativos. . . . .	42



# Capítulo 1

## Introdução

A atividade desportiva de um atleta, numa determinada modalidade, deve ser monitorizada de uma perspetiva externa, no sentido de produzir melhores resultados. Para esse efeito, é necessário que o atleta seja supervisionado durante a sua atividade física, ou que o seu treino seja registado (em papel ou vídeo) e posteriormente analisado.

Neste trabalho, a anotação de eventos sonoros em vídeo refere-se à análise do áudio extraído de um vídeo previamente gravado, com o objetivo de identificar cada instante em que ocorre uma batida de bola (tipo ténis) em raquete. O processamento sobre o áudio será realizado em modo *offline*.

Entre os vários desportos praticados, este trabalho foca-se na análise da componente desportiva de padel ([Courel-Ibáñez et al., 2019]), no sentido de fornecer a atletas e/ou treinadores um complemento no decurso do treino. Uma figura referente ao campo onde este desporto é praticado e à raquete e bola utilizadas encontra-se disponível no Capítulo A, na qual se observam as dimensões do campo onde esta modalidade desportiva é realizada. O desporto de padel pode ser praticado em ambiente interior (*indoor*) ou exterior (*outdoor*). Este trabalho analisa apenas vídeos em ambiente *indoor*.

### 1.1 Motivação

A aplicação em desenvolvimento implica que se implemente um sistema onde a máquina é capaz de reconhecer batidas de bola. No entanto, a forma como uma máquina reconhece um som difere de como esse mesmo som é percebido pelo ouvido humano. O ouvido humano consegue fazer a distinção

entre sons, considerando as características dos sons emitidos: duração, intensidade, tom, entre outras [Council et al., 2004]. É necessário proporcionar à máquina a capacidade de reconhecer batidas de bola de forma semelhante à humana. Para esse efeito, será utilizada a aprendizagem automática.

A aprendizagem automática permite identificar e reconhecer padrões identificativos de batidas de bola no *dataset* fornecido.

A aplicação em desenvolvimento disponibiliza ainda uma interface, que permite que o utilizador participe no processo de anotação dos eventos (validando os resultados devolvidos pelo classificador), o que poderá contribuir numa perspetiva futura, para aumentar o *dataset* e melhorar deteção de batidas de bola.

## 1.2 Principais Contributos

A realização deste trabalho terá os seguintes contributos:

- Não existindo dados relativos a batidas de bola ou que estejam etiquetados nas condições desejadas, será construído um *dataset* de raiz. Os dados constituintes do *dataset* serão obtidos de vídeos amadores;
- Providenciar de um modelo que reconhece batidas de bola;
- Proporcionar de uma aplicação *web* que permite aumentar o *dataset*.

## 1.3 Estrutura do Relatório

O relatório está organizado da seguinte forma:

- O Capítulo 1 introduz a necessidade do uso da aprendizagem automática para reconhecer batidas de bola em vídeos de treinos de padel, enuncia os principais contributos da aplicação em desenvolvimento, e realiza uma apreciação global dos resultados obtidos.
- O Capítulo 2 introduz os conceitos nos quais o trabalho em desenvolvimento se baseia: conceitos sobre processamento de sinal e de aprendizagem automática.



- 
- O Capítulo 3 faz referência a trabalhos relacionados com o trabalho corrente. Estes trabalhos foram sendo analisados durante o desenvolvimento do trabalho e baseiam-se também no uso de técnicas de aprendizagem automática para reconhecer eventos em modalidades desportivas.
  - O Capítulo 4 visa apresentar uma análise geral dos requisitos do sistema e a abordagem utilizada para satisfazer esses requisitos.
  - O Capítulo 5 descreve todo o processo de concretização da solução abordada no capítulo anterior.
  - O Capítulo 6 apresenta os resultados obtidos da implementação da solução adotada para construir o classificador. Neste capítulo é escolhido o modelo (final) que produz os melhores resultados na deteção de batidas de bola.
  - O Capítulo 7 apresenta uma análise geral de todo o trabalho realizado e descreve quais os objetivos a realizar num trabalho futuro.



# Capítulo 2

## Fundamentos

Neste capítulo serão abordados os conceitos de carácter teórico, que sustentam o trabalho realizado.

### 2.1 Conceitos de Processamento de Sinal

Um sinal corresponde a uma grandeza física que varia ao longo do tempo. Em termos matemáticos, um sinal varia em função de uma ou mais variáveis, que podem ser contínuas (sinais contínuos) ou discretas (sinais discretos) [Sampaio et al., 2006].

Neste trabalho, o áudio será extraído do vídeo através de um processo de amostragem e por isso corresponde a um sinal discreto no tempo.

#### 2.1.1 Amostragem

O processo de amostragem converte um sinal contínuo num sinal discreto. Para isso, escolhe pontos equi-espaçados do sinal contínuo para gerar os pontos do sinal discreto. A amostragem de um sinal contínuo com um período específico permite obter um sinal discretizado no tempo [LibreTexts, 2022a]. Cada um dos pontos resultantes é designado por amostra.

O intervalo entre as amostras consecutivas denomina-se período de amostragem e é dado em segundos(s). O inverso deste valor corresponde à frequência de amostragem, que é dada em *Hertz*(Hz), e refere-se ao número de amostras que ocorrem no sinal a cada segundo. A informação removida pelo processo de amostragem pode ser parcialmente recuperada através de interpolação ou reconstrução do sinal [LibreTexts, 2022b].

O ouvido humano deteta sons na gama de frequências compreendida entre 20Hz e 20kHz, pelo que, de acordo com o teorema de *Nyquist* [Evia, 2022], é necessário que a frequência de amostragem dos sons que chegam ao aparelho auditivo humano sejam amostrados a uma frequência duas vezes igual ou superior a 20kHz.

A frequência de amostragem a ser considerada neste projeto é de 44100Hz (cerca de 43kHz), que é um dos valores padrão utilizados [Brown, 2019].

### 2.1.2 Transformada de Fourier

A transformada de Fourier (*Fourier Transform* – FT) é uma função matemática que decompõe um sinal nas suas várias frequências (sinusóides) constituintes [Semmlow, 2012]. O sinal de entrada pertence ao domínio do tempo, e o sinal de saída pertence ao domínio da frequência. Esta função (nos domínios do tempo e frequência) é designada por espectro.

A transformada de *Fourier* é útil para verificar, por exemplo, em que zonas do espectro existe mais ou menos energia concentrada e será utilizada no cálculo das características a extrair do áudio.

### 2.1.3 Características extraídas do áudio

Neste trabalho, a partir do áudio são extraídas as seguintes características: deteção do início de um som (*Onset*) e o valor eficaz (RMS).

O *Onset* divide-se em duas componentes: a deteção do início (*Onset Detect*) e o fluxo espectral (*Spectral Flux*). A primeira componente, corresponde a detetar o instante em que se inicia um determinado som [Rosão, 2012]. A segunda corresponde a verificar variações no espectro do sinal, no sentido de encontrar diferenças entre *frames* consecutivas, o que permite detetar também o início de um som. O fluxo espectral pode também ser definido como uma medida do quão depressa o espectro de um sinal varia [Meyda, 2022].

O valor eficaz (*Root Mean Square* – RMS) refere-se à energia (intensidade) média concentrada numa *frame* [Room, 2021].

A figura 2.1 constitui uma representação das três características enunciadas.

Observando a figura, verifica-se que os picos das duas componentes do *Onset*, bem como o RMS correspondem aos instantes no áudio original onde

a amplitude é maior.

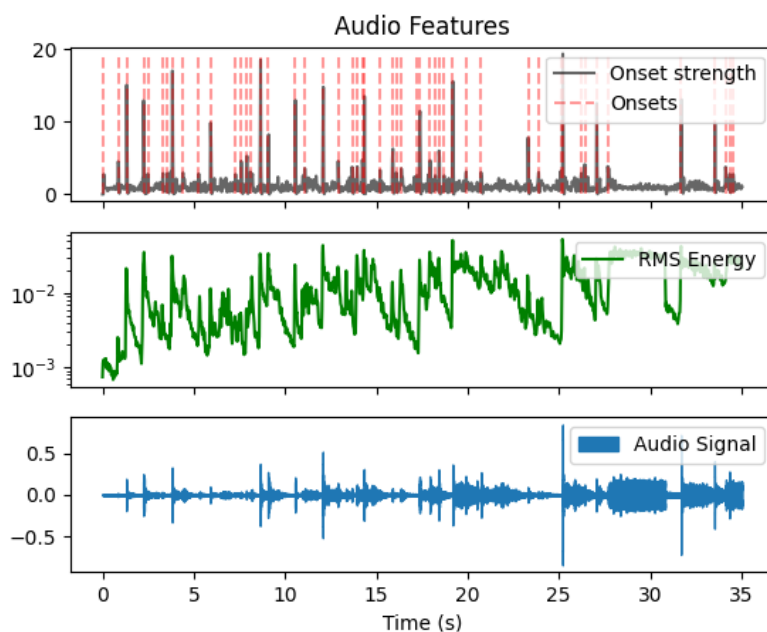


Figura 2.1: Características (cima e meio) e áudio (baixo) correspondente.

As características enunciadas acima podem ser calculadas com o auxílio da biblioteca *librosa* [McFee et al., 2015]. Este módulo realiza processamento sobre sinais de áudio, no sentido de recolher informações sobre o mesmo.

## 2.2 Conceitos de Aprendizagem Automática

A aprendizagem automática (*Machine Learning* - ML) é ramo da inteligência artificial que recorre à análise de (grandes volumes de) dados visando extrair padrões e representá-los num modelo [Janiesch et al., 2021].

Existem três tipos de aprendizagem automática [Sah, 2020]:

- Supervisionada – O modelo é construído com base num conjunto de dados (*dataset*) que associa uma lista de características (*features*) à classe correspondente;
- Não Supervisionada – O modelo é construído com base num conjunto de dados (*dataset*) que apenas contém as características;

- Por Reforço – O modelo é contruído com base na interação com um sistema que gera os dados necessários à extração de padrões.

Este trabalho utiliza aprendizagem supervisionada para extrair os padrões de áudio (e.g., batida de bola em raquete).

### 2.2.1 *Dataset*

Um *dataset* é um conjunto de dados usado no processo de construção dos modelos gerados através de métodos de aprendizagem automática. O *dataset* é estruturado em colunas e linhas (formato de tabela). No caso da aprendizagem supervisionada as colunas representam as características (*features*) e a classe. As linhas representam sempre instâncias (exemplos ou observações).

Considerando uma matriz de características,  $X$ , e o vetor de valores de classes correspondente,  $y$ , o problema de classificação em questão tem como objetivo obter a função,  $f$ , tal que:

$$f(X) = y \quad (2.1)$$

A função  $f$  corresponde a um modelo que deve receber a matriz de características e, classificar de maneira correta cada um dos seus exemplos.

### 2.2.2 *Desiquilíbrio no Dataset*

O desequilíbrio no *dataset* ocorre quando a distribuição de exemplos pelas classes é desigual [Badr, 2019]. Esse desequilíbrio pode resultar na construção de um modelo enviesado que reconhece melhor determinadas classes.

Num problema de classificação binária o desequilíbrio pode abordar-se aumentando o número de exemplos da classe em minoria (sobre-amostragem) ou reduzindo o número da classe em maioria (sub-amostragem) [Badr, 2019].

Em jogos de padel, o número de batidas de bola é muito inferior ao ruído de fundo, pelo que neste trabalho será utilizada a técnica de sub-amostragem para equilibrar o *dataset*.

### 2.2.3 Redes Neurais

Uma rede neuronal artificial (*Artificial Neural Network* – ANN) é uma estrutura usada em métodos de aprendizagem automática que tem como inspiração a biologia humana e a forma como os neurónios comunicam entre si para compreender os dados percecionados [Rauber, 2022].

Uma rede neuronal está organizada em camadas, que são constituídas por unidades (nós). A primeira camada (de *input*), tem um número de nós igual ao número de características do *dataset*. A última camada (de *output*), possui os nós que são usados para representar os valores da classe. Entre a camada de *input* e a camada de *output* existem uma ou mais camadas escondidas. A figura 2.2 representa de uma rede neuronal, onde estão delimitadas as ligações entre os vários nós constituintes.

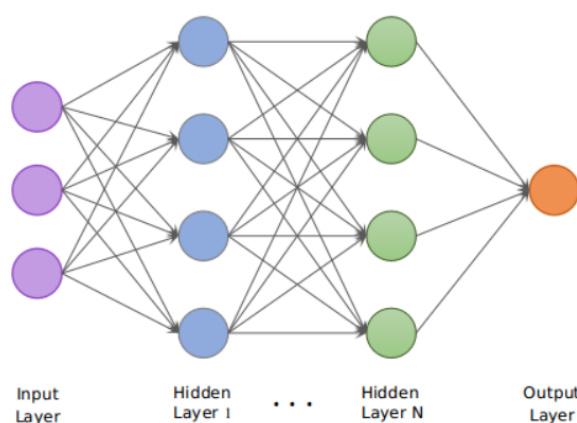


Figura 2.2: Representação típica de uma rede neuronal.

O número de camadas, bem como o número de nós em cada camada são definidos com o objetivo de obter o melhor classificador possível [Krishnan, 2021].

Tal como se verifica na figura 2.2, cada nó recebe os valores de saída dos nós da camada anterior e associa-lhes um peso que é atualizado ao longo do processo de aprendizagem [Medium, 2019].

Os valores de saída são calculados com base numa função de ativação que define a forma como os pesos a serem atualizados são calculados numa determinada camada da rede. Neste trabalho serão utilizadas as funções de ativação [Brownlee, 2021]:

- ReLU (*Rectified Linear Activation*) – recebe um valor e devolve outro

de acordo com a expressão 2.2. Pelo que valores negativos são mapeados no valor 0. Esta função será utilizada nas camadas escondidas.

$$outputValue = \max(0, inputValue) \quad (2.2)$$

- Sigmóide (*Sigmoid*) – recebe um valor e devolve outro no intervalo  $[0, 1]$  de acordo com a expressão 2.3. Esta função será utilizada na camada de *output*.

$$outputValue = \frac{1}{1 + e^{-inputValue}} \quad (2.3)$$

O processo de aprendizagem é realizado ao longo de várias épocas. Uma época (*epoch*) corresponde a um ciclo de passagem, pela rede neuronal, de todos os exemplos do *dataset*. Uma época é constituída por várias iterações. Cada iteração corresponde a uma atualização de pesos após a passagem pela rede de um número pré-definido (*batch*) de exemplos. Um valor de *batch* igual a uma potência de 2 acelera o processo de treino [InterviewArea, 2022].

O uso de métodos de otimização permite atualizar os pesos e a taxa de aprendizagem da rede neuronal de forma a reduzir o erro associado e aumentar a taxa de sucesso. Neste trabalho, será utilizado o método de otimização *Adam* [Doshi, 2019].

### 2.2.4 Hiper-parâmetros

Em aprendizagem automática, um hiper-parâmetro é um valor utilizado pelo modelo para controlar o processo de treino. Os hiper-parâmetros de um modelo não são utilizados diretamente no modelo, mas determinam os parâmetros internos utilizados diretamente. O prefixo “hiper” advém do facto de estes parâmetros serem os escolhidos para definir outros, sugerindo que estão num nível hierárquico superior [Nyuytiybiy, 2021].

### 2.2.5 Sobre-aprendizagem

A sobre-aprendizagem (ou *overfitting*) ocorre quando um modelo se ajusta excessivamente aos dados utilizados para treino. Este ajuste faz com que o modelo produza bons resultados no processo de aprendizagem (conjunto de



treino), e apresente um mau desempenho quando avalia dados novos (conjunto de teste). A sobre-aprendizagem contribui neste sentido para diminuir o desempenho do modelo, pois diminui a capacidade deste de generalizar, isto é classificar corretamente dados nunca antes vistos.

Entre as várias formas de abordar este problema, neste trabalho serão utilizadas as seguintes técnicas:

- Simplificação do modelo – consiste em reduzir a complexidade do modelo. Este processo pode ser realizado reduzindo a quantidade de características. No caso das redes neurais corresponde a reduzir o número de camadas ou de neurónios em cada camada.
- Regularização – permite eliminar a informação que reduz o desempenho do modelo. Neste trabalho serão considerados três tipos de regularização: *Dropout*, *Lasso* e *Ridge*. No método de *Dropout*, a cada iteração do processo de aprendizagem um número (aleatório) de neurónios é desativado [McNealis, 2020]. As regularizações de *Lasso* (ou norma L1) e *Ridge* (ou norma L2) penalizam os pesos associados a certas características. A norma L1 coloca os pesos a zero e a norma L2 diminui os valores dos pesos, evitando que o modelo considere que certas *features* são mais importantes em relação a outras [Pierre, 2021].

### 2.2.6 Validação Cruzada

No caso da aprendizagem supervisionada, a classificação é utilizada para atribuir uma classe (categoria) a um conjunto de exemplos. Após este processo, é construído um modelo que através dos padrões presentes nos dados de *input*, prevê a classe à qual esses dados pertencem. Existem três tipos de classificação [Brownlee, 2022]:

- Multi-Classe – quando existem mais do que duas classes nos dados. Um exemplo não pode pertencer, simultaneamente, a mais do que uma classe.
- Binária – quando existem apenas duas classes nos dados. Um exemplo não pode pertencer, simultaneamente, a mais do que uma classe.

- Multi-Label – quando existem duas ou mais classes e um exemplo pode pertencer, simultaneamente, a mais do que uma classe.

Neste trabalho, o problema de classificação é binário, uma vez que o modelo deve distinguir exemplos que são batidas de bola em raquete (positivos) de exemplos que não são batidas de bola (negativos).

Para avaliar o desempenho de um classificador é necessário verificar quais são as hipóteses de este acertar independentemente da classe, mas também verificar a distribuição de acertos em cada classe. Essas probabilidades podem ser obtidas através da matriz de confusão. Neste trabalho, entre as várias métricas de desempenho que podem ser obtidas da matriz de confusão, serão consideradas as seguintes [Brownlee, 2020]:

- Taxa de sucesso (*Accuracy*) – percentagem de exemplos bem classificados independentemente da classe;
- Precisão (*Precision*) – percentagem de exemplos que o classificador indicou serem de uma determinada classe e que pertencem a essa classe;
- Cobertura (*Recall*) – percentagem de exemplos que pertencem a uma determinada classe e o classificador indicou como sendo dessa classe;
- *F1-score* – corresponde à média harmónica entre a precisão e a cobertura [MachineLearning, 2017].

Em geral, num problema de classificação os dados são divididos em treino e teste. O grupo de exemplos de treino é utilizado no processo de aprendizagem, e o grupo de teste é utilizado no processo de classificação. Embora esta abordagem permita verificar como o classificador se comporta na predição sobre dados novos, tem como desvantagem a hipótese de existirem exemplos mais fáceis de identificar no conjunto de teste. Como resultado, o desempenho do modelo obtido pode não corresponder à realidade. Para contornar este problema pode ser utilizado o método de validação cruzada.

A validação cruzada é uma técnica que permite avaliar a capacidade de generalização de um classificador. Os dados são divididos em  $K$  partições (*folds*) e ao longo de  $K$  iterações são treinados modelos diferentes, onde  $K-1$  itfolds são utilizados para treino e um *fold* é utilizado para validação [Alhamid, 2020].

Dividir os dados em várias partições pode não ser suficiente, uma vez que a distribuição dos exemplos de cada classe pelos *folds* pode não ser igual à original.

De forma a manter essa distribuição de exemplos de cada classe nos *folds* será utilizada a técnica de validação, partição estratificada (*Stratified K-Fold*) [Muralidhar, 2021].



# Capítulo 3

## Trabalho Relacionado

Este capítulo descreve trabalhos relacionados com o tema deste projeto que consiste no uso de aprendizagem automática na detecção de eventos.

### 3.1 Anotação de eventos

O sistema desenvolvido em [Baughman et al., 2019], foca-se na componente acústica para fazer a distinção entre 6 tipos de eventos em jogos de ténis através de uma rede neuronal convulocional (CNN). O sistema proporciona ainda uma ferramenta que permite realizar a anotação de eventos, aumentar o *dataset* e melhorar a qualidade do modelo.

Para poupar tempo no processo de anotação e etiquetação, o sistema proposto em [Kim e Pardo, 2018] faz o reconhecimento (em áudio) de eventos semelhantes a um evento fornecido pelo utilizador. Este evento fornecido pode ser uma região do áudio selecionada pelo utilizador. O áudio é dividido em pequenos segmentos do mesmo tamanho que o evento que se pretende identificar. O sistema seleciona os segmentos que possuem as características que se aproximam mais do evento em análise e mostra os resultados através de uma interface. O utilizador pode validar e realizar ajustes nos resultados obtidos, o que contribui para melhorar o processo de classificação do modelo utilizado.

O sistema apresentado em [Liu et al., 2006] realiza a detecção de eventos combinando informação visual e acústica obtida em vídeos de basquetebol. A componente acústica foca-se nos eventos que poderão ser indicadores dos momentos mais importantes no jogo, como o som emitido pelos espetadores

ou o entusiasmo na voz de um comentador.

## 3.2 Extração de Características e Reconhecimento de Padrões

Ainda no sistema apresentado em [Baughman et al., 2019], o sinal de áudio é dividido em *frames* de 20 milissegundos a partir das quais se obtém os coeficientes de MFCC para reconhecer os eventos de forma semelhante à humana. O sistema proposto em [Liu et al., 2006] considera também o nível de energia associado aos segmentos no processo de construção dos vetores de características.

O sistema descrito em [Ganser et al., 2021] permite recolher informação relativa a batidas de bola na atividade desportiva de ténis através de sensores em pulseiras eletrónicas e uma câmara de vídeo. A pulseira grava cerca de 11 sinais, incluindo a unidade de medida de inercial (IMU). No processo de etiquetação a informação obtida em vídeo é sincronizada e combinada com os sinais gravados na pulseira eletrónica para construir o *dataset*. O processo de classificação é realizado através das CNNs.

# Capítulo 4

## Modelo Proposto

Neste capítulo será realizada uma descrição geral do sistema desenvolvido, dos requisitos e da abordagem adotada para o desenvolver.

### 4.1 Descrição Geral do Sistema

O sistema desenvolvido reconhece batidas de bola, permite auxiliar no processo de construção do *dataset* e, conseqüentemente no processo de melhoria de desempenho do classificador. O funcionamento do sistema pode ser representado como se mostra na figura 4.1.

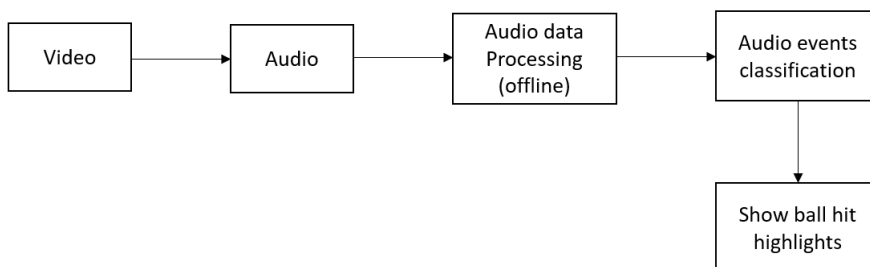


Figura 4.1: Sistema de anotação de eventos desejado.

### 4.2 Requisitos

Os requisitos do sistema podem ser categorizados como funcionais (funções de sistema) e não funcionais (atributos do sistema). As funções de sistema,

referem-se a funcionalidades desempenhadas pelo sistema. Os atributos do sistema descrevem a forma como o sistema deve ser disponibilizado.

Neste projeto, os requisitos funcionais foram organizados em dois grupos:

- Construção do *dataset*;
- Construção do classificador.

A tabela 4.1 contém os requisitos funcionais que dizem respeito à criação do *dataset*:

Requisito	Função
R1.1	Extrair o áudio do vídeo
R1.2	Anotar as batidas de bola presentes no áudio (A)
R1.3	Extrair as características do áudio (B)
R1.4	Combinação dos dois conjuntos A e B

Tabela 4.1: Requisitos funcionais do sistema em desenvolvimento para a construção do *dataset*.

A tabela 4.2 contém os requisitos funcionais que dizem respeito ao processo de classificação:

Requisito	Função
R1.1	Extrair o áudio do vídeo
R1.2	Extrair as características do áudio
R1.3	Aplicar o classificador
R1.4	Apresentar resultados dos eventos etiquetados

Tabela 4.2: Requisitos funcionais respeitantes ao processo de classificação.

A tabela 4.3 representa os atributos do sistema.

Atributo	Detalhe/ Restrição Fronteira
Interação Homem-Máquina	Detalhe Interface fácil de aprender a usar

Tabela 4.3: Requisitos não funcionais do sistema.



## 4.3 Abordagem

Nesta secção será descrita a abordagem utilizada para realizar cada uma das etapas inerentes ao sistema em desenvolvimento. As etapas são as seguintes:

- Construção do *dataset*;
- Construção de um classificador que reconhece batidas de bola;
- Desenvolvimento de uma aplicação *web* que contribui para aumentar o *dataset*.

### 4.3.1 Construção do *Dataset*

O *dataset* construído corresponde a um conjunto de dados com padrões identificativos de batidas de bola.

#### Obtenção da matriz de características – $X$

O impacto de uma batida de bola tem uma duração média de 0.5 segundos. Por isso, no processo de extração das características (cf., 2.1.3) é realizado um varrimento sobre o áudio com uma janela temporal de dimensão fixa igual a meio segundo. Este processo é realizado ao longo de várias iterações, onde em cada iteração a janela desliza um número específico de amostras e as amostras abranjidas pela janela são utilizadas para o cálculo das características pretendidas.

O número de amostras a deslizar ou o salto (“*hop*”) considerado será de 1024 amostras como primeira abordagem. No entanto, podem ser escolhidos múltiplos deste valor como outras abordagens, no sentido de verificar que alterações produzem no desempenho do classificador (tabela 4.4).

A matriz  $X$  é constituída por  $M$  linhas ou exemplos. Por sua vez, cada exemplo tem  $N$  colunas ou valores de características, pelo que a matriz  $X$  pode ser representada na figura 4.2.

O valor de  $N$  na figura 4.2, corresponde ao número de segmentos de amostras abranjidos pela janela fixa para uma característica e pode ser dado pela seguinte expressão:

$$N = \frac{eventLength \times samplingRate}{hopLength} \quad (4.1)$$

onde na equação, *eventLength* é a duração de um evento, *samplingRate* é a frequência de amostragem (a que os áudios são obtidos), e *hopLength* refere-se ao número de amostras deslizadas a cada iteração do varrimento.

$$X = \begin{bmatrix} X_1 & X_2 & X_3 & \cdots & X_N \\ X_2 & X_3 & X_4 & \cdots & X_{N+1} \\ X_3 & X_4 & X_5 & \cdots & X_{N+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ X_{N-1} & X_N & X_{N+1} & \cdots & X_{2N-2} \\ X_N & X_{N+1} & X_{N+2} & \cdots & X_{2N-1} \\ X_{N+1} & X_{N+2} & X_{N+3} & \cdots & X_{2N} \\ \vdots & \vdots & \vdots & & \vdots \end{bmatrix}$$

Figura 4.2: Matriz de características.

Na tabela 4.4, estão evidenciados os valores de N obtidos para as várias combinações de *eventLength* e *hopLength*:

<i>eventLength</i>	<i>hopLength</i>	<i>N</i>
0.5	1024	22
0.5	2048	11
0.5	4096	5

Tabela 4.4: Valores de duração (em segundos), amostras deslizadas e N a considerar, para um valor de *samplingRate* igual a 44100Hz.

Observando ainda a figura 4.2, a primeira iteração do processo de varrimento permite obter a primeira linha da matriz X. A segunda iteração permite obter a segunda linha da matriz e, assim sucessivamente até à iteração M. Note-se que o valor de N, corresponde ao número de segmentos de amostras obtidos para uma característica. Por isso, caso se pretenda considerar as três características enunciadas na secção 2.1.3, o número de elementos em cada linha da matriz X, é igual a três vezes o valor de N.

### Obtenção do vetor de classes – *y*

Cada um dos exemplos da matriz de características pertence a uma classe ou categoria. O vetor de classes pode ser representado como se mostra na figura

4.3, onde cada elemento  $y_i$  corresponde à classe etiquetada. Para obtenção deste vetor, é necessário etiquetar manualmente o áudio. O processo resume-se a ouvir o áudio e anotar num ficheiro (formato.csv), em que intervalos de tempo ocorrem batidas de bolas.

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_M \end{bmatrix}$$

Figura 4.3: Vetor de classes.

Note-se que apenas as batidas de bola são registadas no ficheiro. Por isso, todos os eventos que não são registados constituem ruído. Cada um dos exemplos,  $y_i$ , do vetor  $y$  terá um valor dependente da expressão 4.2:

$$y_i = \begin{cases} \text{ball-hit} & \forall X_i: X_i \in [\text{beginSample}, \text{endSample}] \\ \text{noise} & \forall X_i: X_i \notin [\text{beginSample}, \text{endSample}] \end{cases} \quad (4.2)$$

onde  $[\text{beginSample}, \text{endSample}]$  é o intervalo de tempo em que a batida de bola ocorre.

### Criação do *dataset*

O processo de construção do *dataset* refere-se à junção da matriz de características ao vetor de classes. A janela desliza ao longo do áudio por isso, um conjunto de exemplos ou instâncias,  $X_i$ , corresponderão a uma classe,  $y_i$ , tal como se verifica na figura 4.4.

O vídeo de *input*, a partir do qual o *dataset* foi construído é um vídeo obtido de forma não profissional com uma duração de 1 hora e 32 minutos. e constitui uma gravação de um treino realizado por um conjunto de atletas em ambiente *indoor*.

$$X = \begin{bmatrix} X_1 & X_2 & X_3 & \cdots & X_N & y_1 \\ X_2 & X_3 & X_4 & \cdots & X_{N+1} & y_2 \\ X_3 & X_4 & X_5 & \cdots & X_{N+2} & y_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ X_{N-1} & X_N & X_{N+1} & \cdots & X_{2N-2} & y_{N-1} \\ X_N & X_{N+1} & X_{N+2} & \cdots & X_{2N-1} & y_N \\ X_{N+1} & X_{N+2} & X_{N+3} & \cdots & X_{2N} & y_{N+1} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \end{bmatrix}$$

Figura 4.4: Matriz de características.

### 4.3.2 Construção do Classificador

A escolha do algoritmo de aprendizagem automática a utilizar foi realizada através da ferramenta *Orange Data Mining* – ODM. Esta ferramenta permite aplicar vários algoritmos aos dados, no sentido de verificar quais destes produzem os melhores resultados [Baptista, 2019]. Para verificar a qualidade dos modelos é utilizada a técnica partição estratificada (*Stratified KFold*) onde os dados são divididos em 5 *folds*.

A construção do classificador envolve também a escolha dos hiper-parâmetros que produzem os melhores resultados.

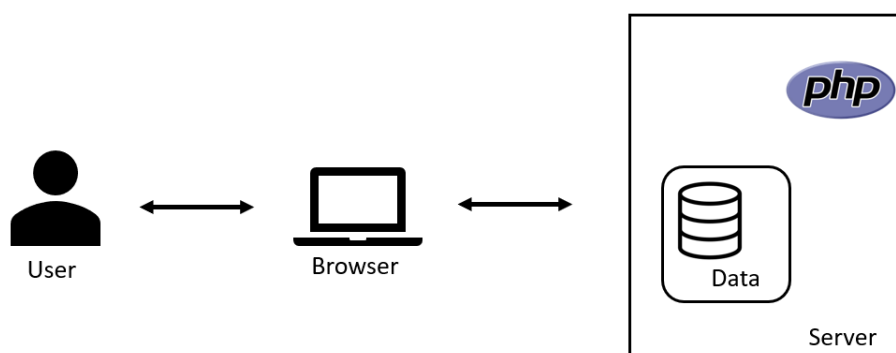
### 4.3.3 Desenvolvimento da Aplicação Web

Esta sub-seção faz uma descrição da arquitetura e o *design* da aplicação que deve mostrar os resultados obtidos pelo classificador.

#### Arquitetura da Aplicação Web

A aplicação divide-se em três componentes: cliente, servidor e dados. A estrutura da aplicação *web* encontra-se na figura 4.5. Estas componentes serão todas integradas na mesma máquina para efeitos de simplificação.

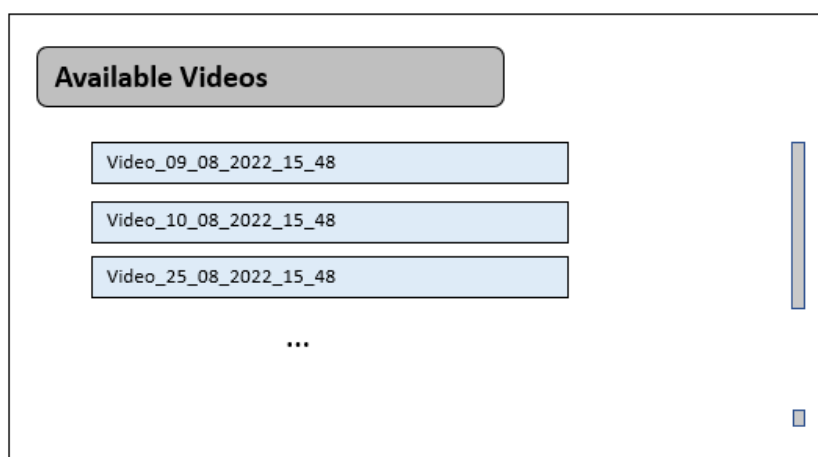
A aplicação em desenvolvimento faz uso da linguagem PHP. Esta linguagem é executada no lado do servidor e permite criar interação com o utilizador através de páginas HTML dinâmicas. No lado do cliente será utilizada a linguagem *Javascript* e AJAX através dos quais serão utilizados os métodos GET e POST para receber ou enviar dados para o servidor [W3schools, 2022].

Figura 4.5: A da aplicação *web*.

### *Design* da Aplicação

A aplicação *web* desenvolvida tem duas páginas: página (1) que permite escolher o vídeo sobre o qual se pretende visualizar os resultados; página (2) que permite visualizar os resultados devolvidos pelo classificador (para um determinado vídeo). O *design* destas páginas encontra-se nas figuras 4.6 e 4.7. Todo o processamento efetuado sobre o áudio é realizado antes de apresentar os resultados nesta aplicação.

Para simplificar, a página 1 terá um lista de vídeos que já foram classificados. Ao selecionar um vídeo é redirecionado para a página 2, que permite analisar os vários *clips* constituintes do vídeo selecionado.

Figura 4.6: *Design* desejado da página *web* 1.

Os *clips* são resultado do processo de deslizamento da janela ao longo do vídeo (de teste) a classificar. Desta forma, cada *clip* tem a duração de 0.5 segundos. Através do botão “*select*” é possível selecionar um *clip* (à esquerda) e visualizá-lo (à direita). Ainda na página 2 o utilizador pode validar os resultados devolvidos pelo classificador alterando o tipo de evento (entre “*ball-hit*” e “*noise*”) na primeira coluna. É ainda possível ordenar e visualizar os eventos por probabilidade de serem batida de bola (na ordem decrescente) clicando no botão “*Sort by probability*”.

O botão “*Submit*” permite enviar as alterações realizadas ao servidor através de um pedido do tipo *POST*. As alterações submetidas são guardadas em ficheiros de texto (.txt) e contêm informação acerca do tipo de evento escolhido pelo utilizador e as amostras inicial e final respetivas. Estas informações são posteriormente utilizadas para aumentar o *dataset* e consequentemente melhorar a qualidade do modelo. Esta abordagem é bastante útil na anotação de eventos para vídeos não anotados.

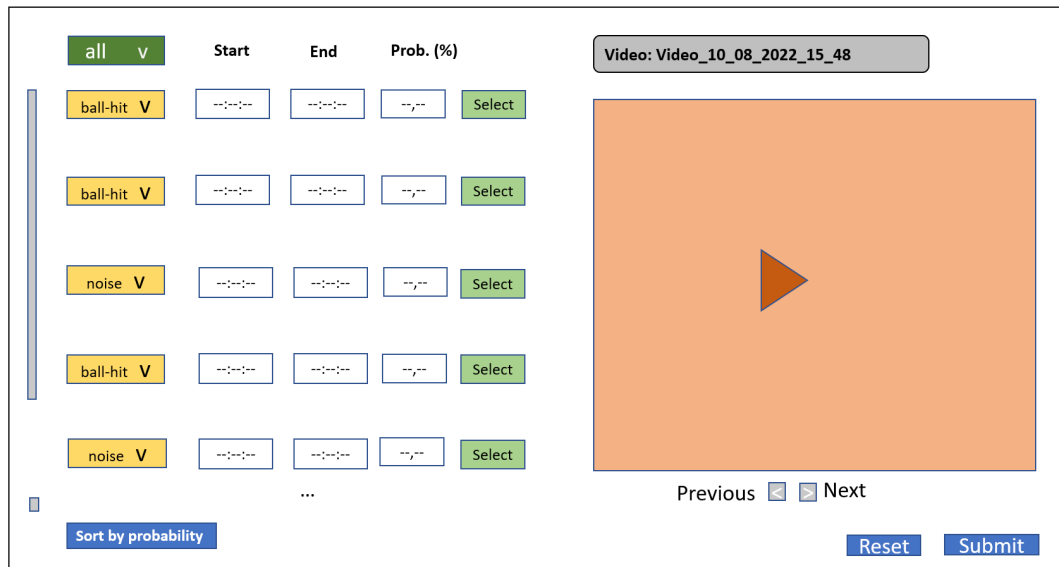


Figura 4.7: *Design* desejado da página *web* 2.

# Capítulo 5

## Implementação do Modelo

Este capítulo descreve todo o processo de implementação das etapas enunciadas na secção abordagem (4.3).

### 5.1 Construção do *Dataset*

O processo de criação do *dataset* pode ser representado pelo esquema que se encontra na figura 5.1.

Para simplificar o processo de anotação das batidas de bola (“*Human ear annotation*”), optou-se por segmentar o vídeo de longa duração em vídeos com 1 minuto. O processo de segmentação deu origem a cerca de 90 vídeos de curta duração. Para cada um desses vídeos, foram gerados os áudios correspondentes e 30 foram anotados.

A anotação das batidas de bola foi realizada com o auxílio da aplicação *Adobe Audition* [Adobe, 2013]. Esta ferramenta permite efetuar uma análise sobre áudio e verificar em que instantes ocorrem batidas de bola. A anotação é realizada num ficheiro *Excel*, que posteriormente é convertido para *CSV*.

Cada evento está registado no ficheiro *Excel* com os seguintes dados:

- O tipo de batida de bola (em raquete, no chão, na rede metálica, entre outros). Todos estes tipos pertencem à classe batida de bola, no entanto apenas as batidas de bola em raquete foram consideradas neste projeto;
- O intervalo de tempo em que ocorre a batida de bola (amostras inicial e final no áudio);
- O tipo de ambiente em que o treino é praticado (*indoor* ou *outdoor*);

- O lado do campo em que batida de bola ocorre.

O tipo de ambiente onde o desporto é praticado e o lado do campo onde ocorrem as batidas de bola estão fora do contexto do presente trabalho.

Na fase de extração de características iterou-se apenas sobre os áudios anotados. Sobre estes áudios realizou-se o processo descrito na secção 4.3.1 para obtenção da matriz de características com o auxílio da biblioteca *tensorflow*. Esta biblioteca permite realizar indexação e varrimento sobre *arrays* de forma simples.

Na fase de “*Labeling*” combinou-se a informação registada nos ficheiros *CSV* (batidas de bola) com as características tal como se verifica na figura 4.4, o que permitiu obter o *dataset* a utilizar na construção do modelo (“*Input data*”).

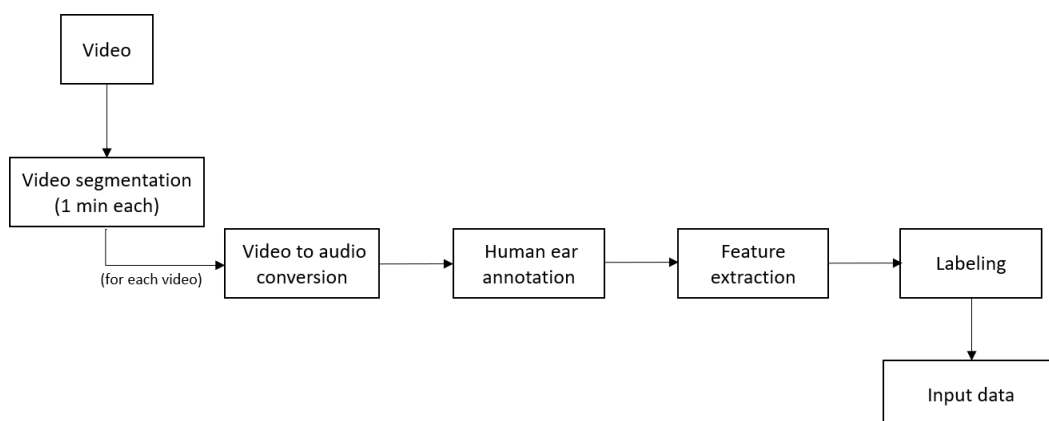


Figura 5.1: Processo detalhado de construção do *dataset*.

Considerou-se como primeira abordagem a primeira linha da tabela 4.4, onde a cada iteração a janela desliza 1024 amostras e abranje 22 grupos de 1024 amostras (0.5 segundos). O *dataset* obtido contém cerca de 75 mil exemplos: 19724 exemplos de batidas de bola e 54646 exemplos de ruído.

## 5.2 Construção do Classificador

A construção do modelo pode ser descrita através da figura 5.2, que descreve de uma forma geral as fases envolvidas neste processo.



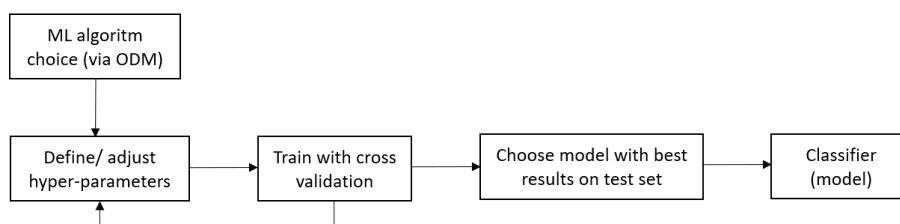


Figura 5.2: Processo de construção do classificador.

### Escolha do modelo

Através da ferramenta ODM aplicaram-se vários algoritmos ao *dataset* e obtiveram-se os resultados que se encontram na figura 5.3. Estes resultados (independentes das classes) sugerem que os modelos com melhores desempenhos são o regressor logístico e a rede neuronal. Para estes dois classificadores os valores das métricas de avaliação (cf., 2.2.6) estão acima dos 85% e o erro (“*LogLoss*”) associado à classificação é igual ou inferior 34%.

As figuras B.2 e B.3 correspondem aos resultados para cada uma das classes (positivos e negativos). Observando separadamente estes resultados, verifica-se que a rede neuronal obtém os melhores resultados em termos de taxa de sucesso, média harmónica e erro associado. Por este motivo, a rede neuronal será a técnica de aprendizagem automática a utilizar para construir o classificador.

Test and Score - Orange

☒ Cross validation  
 Number of folds: 5  
☒ Stratified  
☐ Cross validation by feature  
☐ Random sampling  
 Repeat train/test: 10  
 Training set size: 70 %

Evaluation results for target (None, show average over classes)

Model	CA	F1	Precision	Recall	LogLoss
Neural Network	0.937	0.937	0.937	0.937	0.192
Tree	0.909	0.908	0.908	0.909	1.619
Logistic Regression	0.877	0.869	0.878	0.877	0.339
Naive Bayes	0.836	0.841	0.854	0.836	2.500
SVM	0.435	0.468	0.545	0.435	0.788

Figura 5.3: Resultados (independentes das classes) da aplicação de vários algoritmos aos dados.

De forma obter resultados próximos dos observados para a rede neuronal serão considerados os hiper-parâmetros utilizados na aplicação ODM.

### Definição dos Hiper-parâmetros e Treino do Modelo

Para construir e treinar o modelo utilizou-se a biblioteca *Python Keras*. Esta biblioteca usa internamente o módulo *tensorflow* e permite criar modelos de aprendizagem automática de forma simples e flexível [Keras, 2022].

A capacidade de generalizar do modelo foi verificada através do método de validação cruzada, *Stratified KFold*. Tal como na ferramenta ODM, os dados foram divididos em 5 *folds*.

Os hiper-parâmetros considerados foram os seguintes:

- Temporais – duração da janela de dimensão fixa (*eventLength*) e número de amostras deslizadas a cada iteração (*hopLength*);
- Na rede neuronal – tipo de regularização e o parâmetro *lambda* associado, taxa de aprendizagem, método de otimização, número de épocas, número de *batches*, número de camadas na rede e de neurónios em cada camada.

Os parâmetros temporais têm influência na estrutura do *dataset* por isso também têm impacto na construção do modelo. Os parâmetros da rede neuronal têm impacto na forma como o modelo se ajusta aos dados recebidos.

O objetivo é escolher os hiper-parâmetros do modelo que produzem os melhores resultados. Para esse efeito, após a verificação dos resultados obtidos no treino e validação, os hiper-parâmetros foram ajustados várias vezes, no sentido de melhorar a qualidade do modelo.

## 5.3 Desenvolvimento da Aplicação *web*

Os *clips* constituintes de cada vídeo de teste, bem como todas as informações a eles associadas (eventos em análise, tempos iniciais e finais de cada evento, probabilidade de um evento ser batida de bola, entre outras) foram organizados numa diretoria através de código *Python*. Essa diretoria é posteriormente copiada para a aplicação desenvolvida em PHP.

A aplicação *web* obtém e processa as informações contidas na diretoria e mostra os resultados ao utilizador. O esquema da figura 5.4 descreve o processo de desenvolvimento da aplicação *web*.

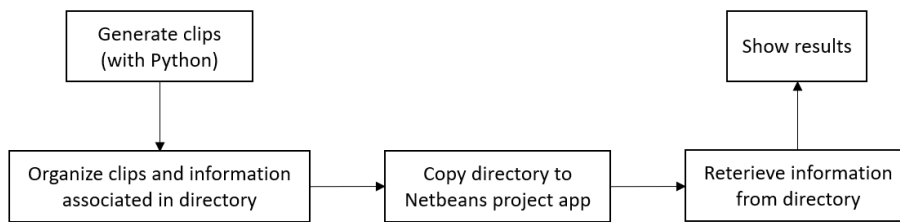


Figura 5.4: Processo de desenvolvimento da aplicação *web*.

A figura 5.5 representa a estrutura da aplicação. A diretoria “*Lib*” contém as constantes, métodos auxiliares (em “*jsScripts.js*” e “*styles.css*”) e uma folha de estilos (CSS). A diretoria “*videos*” recebe outras diretorias que contém informação sobre vídeos classificados. As páginas PHP abaixo permitem apresentar os resultados ao utilizador.

Do ponto de vista do utilizador, quando este submete as alterações efetuadas na interface apresentada na figura 4.7, os ficheiros de texto são guardados na pasta “*userAnalysis*”.

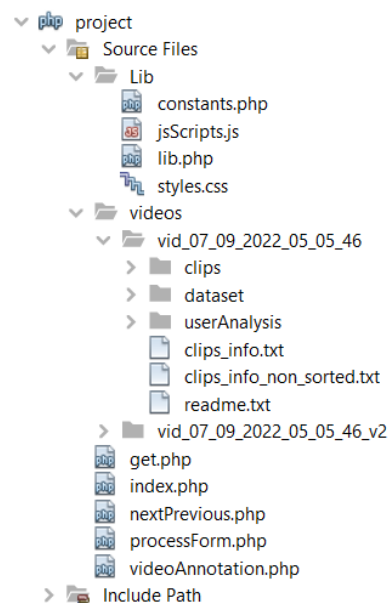


Figura 5.5: Estrutura do projeto correspondente à aplicação *web* no IDE Netbeans.



# Capítulo 6

## Validação e Testes

Neste capítulo serão apresentados os resultados obtidos ao longo do processo de construção do modelo final a ser utilizado para a classificação de batidas de bola.

### 6.1 Definição dos Hiper-parâmetros

Para obter um modelo que identifique corretamente batidas de bola em raquete foram realizados vários experimentos, alterando os parâmetros da rede neuronal. A tabela 6.1 contém os resultados obtidos para esses experimentos. Considerou-se um número de *batches* igual a 128, a função de erro *binary cross-entropy* o algoritmo de otimização *Adam*. Todos os outros hiper-parâmetros encontram-se evidenciados na tabela referida.

A duração dos eventos (*eventLength*) foi pré-definida para meio segundo (0.5 s), por isso, o valor de  $N$  (na expressão 4.1) e o número de exemplos no *dataset* em (“*Dataset Observations*”) dependem do valor do *hopLength* (“*Hop*”). Cada um dos experimentos (1 a 18) foi realizado aplicando a técnica *Stratified KFold*, o que permitiu obter 5 modelos diferentes. Os resultados presentes na tabela 6.1 constituem os valores médios obtidos para modelos construídos. No caso do experimento 1, tem-se o seguinte:

- O *dataset* utilizado foi construído considerando um *hopLength* igual a 1024. A proporção entre batidas de bola e ruído não é igual, uma vez que existem mais exemplos de ruído do que de batidas de bola.
- Os modelos resultantes da validação cruzada têm 2 camadas escondidas

(com 64 e 32 neurónios cada), sobre as quais são aplicadas as técnicas de regularização *dropout* e normas L1 e L2;

- Cada um dos modelos foi treinado ao longo de 100 épocas com o conjunto de treino e testado com o conjunto de validação;

O experimento 2 difere do experimento 1 apenas pelo *dataset* que está equilibrado. Os resultados médios de *accuracy* e *loss* destes dois experimentos estão bastante próximos tanto para conjunto de treino como para o conjunto de validação. Num dos modelos obtidos para o experimento 1, verifica-se uma média harmónica (*f1-score*) de 87% para a classe dos positivos e 96% para a classe dos negativos. A taxa de acertos é de 93% e o erro associado ao processo de aprendizagem é de 21%. Embora os resultados sejam bons, o desequilíbrio leva a que o modelo classifique com melhor desempenho os exemplos da classes dos negativos. O mesmo acontece para os modelos dos experimentos 4 e 8.

Nos experimentos 3 a 9 (inclusive) procurou-se verificar qual o impacto da ausência da regularização através da norma L1 no desempenho dos modelos. No entanto, não se verificaram diferenças significativas nos resultados (em relação aos outros experimentos).

Nos experimentos 10 a 12 (inclusive) verificou-se que, independentemente do valor de *hopLength*, o uso método *dropout* permite obter resultados tão bons quanto nos casos em que se usam somente as outras técnicas de regularização (L1 ou L2) ou uma combinação das três técnicas.

Os experimentos 13 a 18 foram realizados com o mesmo *dataset*, tendo-se obtido a melhor taxa de acertos para o experimento 17.

Dos experimentos em análise, foi escolhido o experimento 12, uma vez que a rede neuronal correspondente é a mais simples comparando com as restantes e o valor de *hopLength* (4096) implica menos anotações por parte do utilizador. No entanto, o modelo foi treinado e testado com vídeos gravados nas mesmas condições. Para verificar como o modelo se comporta em relação a vídeos gravados em condições diferentes, construiu-se um modelo com os mesmos hiper-parâmetros do experimento 12. De seguida, treinou-se esse modelo utilizando todos os dados do *dataset*.

Experiment	Hop	Layers	Neurons		Regularization			Epochs	Dataset Observations			
			1st Layer	Hidden Layers	Dropout	L1	L2		Train		Validation	
									Ball Hits	Noise	Ball Hits	Noise
1	1024	4	66	64	0.4	10-3	10-3	100	15780	43716	3944	10930
				32	0.2							
2	1024	4	66	64	0.4	10-3	10-3	100	12653	12739	3163	3185
				32	0.2							
3	2048	3	33	256	0.45	N/A	10-5	100	6328	6367	1582	1592
4	2048	3	33	256	0.45	N/A	10-5	300	7897	21863	1974	5466
5	2048	3	33	256	0.45	N/A	10-5	300	6328	6367	1582	1592
6	4096	4	15	32	0.4	N/A	10-5	300	3068	3213	767	804
				16	0.2							
7	4096	3	15	64	0.4	N/A	10-4	200	3068	3213	767	804
8	4096	3	15	64	0.4	N/A	10-4	200	11076	3828	958	2768
9	1024	3	66	64	0.4	N/A	10-6	200	12653	12739	3163	3185
10	1024	3	66	64	0.4	N/A	N/A	200				
11	2048	3	33	64	0.4	N/A	N/A	200	6328	6367	1582	1592
12	4096	3	15	64	0.4	N/A	N/A	200	3068	3213	767	804
13	1024	4	66	64	N/A	10-2	N/A	200	12653	12739	3163	3185
				32								
14	1024	4	66	64	N/A	10-4	N/A	200				
				32								
15	1024	3	66	128	N/A	10-3	10-3	200				
16	1024	4	66	32	N/A	N/A	10-5	200				
				32								
17	1024	3	66	128	N/A	N/A	10-5	200				
				32								
18	1024	3	66	64	N/A	N/A	10-5	200				

Experiment	Learning Rate	Accuracy		Loss		Activation Function
		Train	Validation	Train	Validation	
1	10-3	92%	92%	0.25	0.24	sigmoid
2		92%	93%	0.25	0.23	
3		93%	92%	0.18	0.21	
4		95%	94%	0.16	0.18	
5		94%	92%	0.16	0.22	
6		91%	90%	0.25	0.26	
7	10-2	90%	90%	0.27	0.28	
8		92%	93%	0.23	0.22	
9	10-4	93%	93%	0.19	0.19	
10	10-3	94%	93%	0.16	0.18	
11		93%	92%	0.19	0.21	
12		91%	90%	0.25	0.26	
13		89%	89%	0.32	0.32	
14		94%	93%	0.18	0.22	
15		95%	93%	0.15	0.22	
16		95%	93%	0.13	0.21	
17		97%	93%	0.09	0.25	
18		96%	93%	0.11	0.23	

Tabela 6.1: Resultados obtidos para os vários hiper-parâmetros da rede neuronal e *datasets* diferentes.

## 6.2 Classificação de Vídeos Novos

Para verificar a capacidade de generalizar do classificador escolhido, etiquetaram-se mais três vídeos (A, B e C). Estes vídeos (de qualidade profissional) têm durações entre 25 e 45 segundos.

Ao classificar os vídeos referidos acima obtiveram-se os resultados representados na tabela 6.2. Estes resultados sugerem que para estes vídeos, o erro associado à classificação é mais alto e a taxa de acertos é mais baixa. Facto que pode ser explicado pela diferença nas condições em que os vídeos foram gravados.

O classificador obteve o menor desempenho para o vídeo B. Observando a a matriz de confusão na figura 6.1, verifica-se que este desempenho se deve ao número elevado de falsos negativos. O classificador obteve o melhor desempenho para o vídeo A. A matriz de confusão respetiva permite verificar que o classificador consegue identificar corretamente grande parte dos eventos.

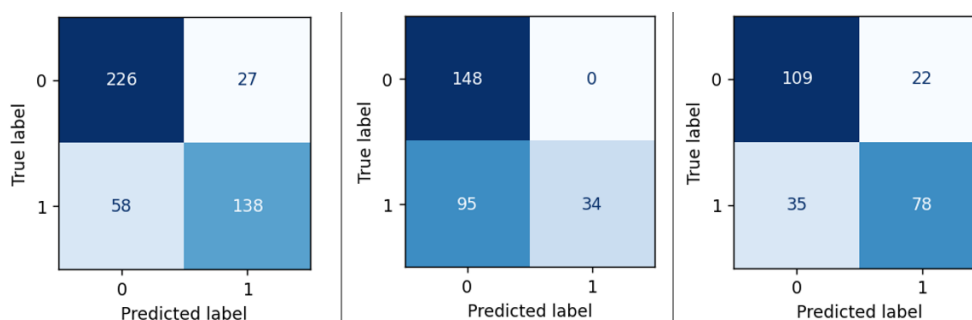


Figura 6.1: Matrizes de confusão resultantes da classificação dos vídeos A (esquerda), B (meio) e C (direita).

Test Accuracy			Test Loss		
Video A	Video B	Video C	Video A	Video B	Video C
81%	66%	0.77	0.42	1.17	0.53

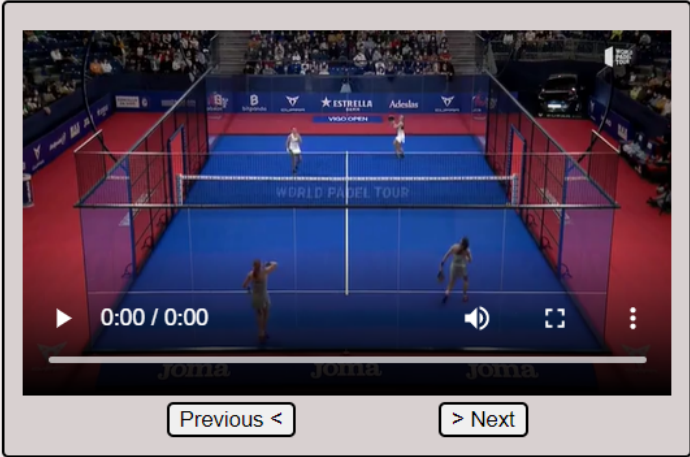
Tabela 6.2: Resultados da classificação sobre vídeos com características diferentes.



## 6.3 Anotação e Melhoria do *Dataset*

Para que o classificador consiga detetar outros “tipos” de batidas de bola em raquete é necessário alargar o *dataset* adicionando mais exemplos com esses eventos. Para esse efeito, anotou-se o vídeo A através da aplicação *web* desenvolvida. A interface utilizada para realizar a anotação encontra-se na figura 6.2.

Video in Analysis: vid\_09\_09\_2022\_05\_06\_41



all	Start Time	End Time	Prob.(%)	
noise	00:00:06.687	00:00:07.151	0.187	Select
noise				
ball-hit	00:00:06.780	00:00:07.244	0.047	Select
noise				
noise	00:00:06.873	00:00:07.337	0.068	Select

Sort by probability Submit Reset

Figura 6.2: Interface da página 2 da aplicação *web*.

O ficheiro de texto obtido das alterações submetidas foi utilizado para estender o *dataset* e construir um novo modelo, agora com mais informação para identificar novos tipos de batidas de bola em raquete.



# Capítulo 7

## Conclusões e Trabalho Futuro

### 7.1 Conclusões

Neste projeto foi desenvolvida uma ferramenta de anotação que permite identificar batidas de bola (em raquete) em jogos de padel. A anotação manual de eventos constitui um processo bastante demorado por isso a aplicação surgiu no sentido de contornar esse problema.

No processo de construção do *dataset*, a tarefa de criação da janela deslizante permitiu definir vários valores para os saltos que ocorrem a cada iteração no processo de varrimento. Verificou-se também que saltos diferentes produzem *datasets* diferentes, o que permite criar variabilidade no dados utilizados para treinar a rede neuronal.

Os testes realizados variando os hiper-parâmetros na rede neuronal permitiram verificar que os *datasets* desequilibrados levaram à construção de modelos que reconhecem melhor os exemplos da classe dos ruído do que os exemplos da classe da batida de bola. Verificou-se também que a utilização da técnica de regularização *Dropout* é bastante eficaz na redução da complexidade do modelo, permitindo que este generalize com mais facilidade.

A escolha do modelo primou pela simplicidade e pela redução do tempo a ser dispendido pelo utilizador no decurso da anotação automática de eventos.

Este projeto permite concluir que é possível utilizar técnicas de aprendizagem automática na deteção de eventos em jogos de padel.

## 7.2 Trabalho Futuro

A aplicação desenvolvida já identifica algumas batidas de bola, mas neste momento o *dataset* está bastante incompleto em termos de variabilidade. Neste sentido, seria bastante vantajoso utilizar a ferramenta para ir aumentando o *dataset* de forma automática.

As características utilizadas para extrair informação do som podem ser revistas, no sentido de se verificar se existem características que permitam distinguir com maior facilidade os eventos em análise.

Relativamente à construção do modelo poderiam ser utilizados outros algoritmos como por exemplo o regressor logístico, que de acordo com a aplicação ODM, também classifica as batidas de bola com alguma robustez.

A adição de outros tipos de eventos, isto é o reconhecimento de outros tipos de eventos ou outros tipos de batidas de bola também poderia contribuir para tornar a aplicação mais rica.

Espera-se, numa perspetiva futura, que esta aplicação seja útil no reconhecimento de batidas de bola.

# Apêndice A

## Um Detalhe Adicional

Durante o processo de desenvolvimento deste projeto, foi utilizado um sistema de gestão de versões para poder guardar todo o progresso do trabalho. Este sistema de gestão permitiu guardar e recuperar as várias versões que foram sendo desenvolvidas ao longo do trabalho realizado. Entre os vários documentos guardados nas versões salienta-se o relatório, código desenvolvido e outras ideias ou rascunhos de útil análise.

Foi utilizado o *Git* para fazer a interligação entre o repositório local e o repositório na *Internet*. O projeto foi realizado individualmente, por isso, foi apenas utilizado um ramo (*branch*) no processo de gestão de versões.



## Apêndice B

### Outro Detalhe Adicional

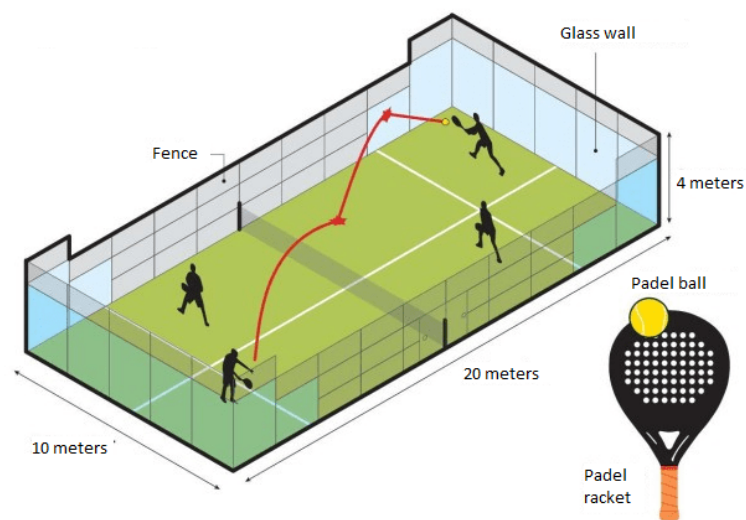


Figura B.1: Descrição pormenorizada do campo de pádel. [Tennisnerd, 2015]

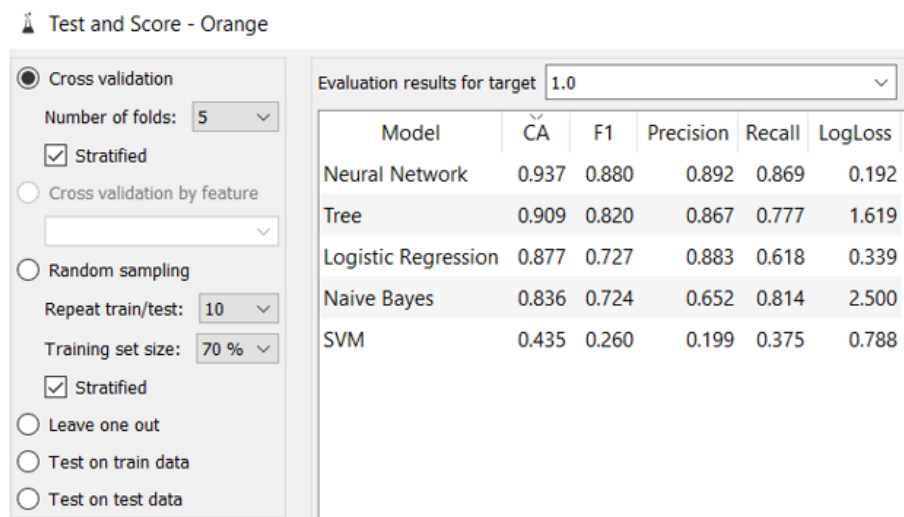


Figura B.2: Resultados da aplicação de vários algoritmos aos dados para a classe dos positivos.

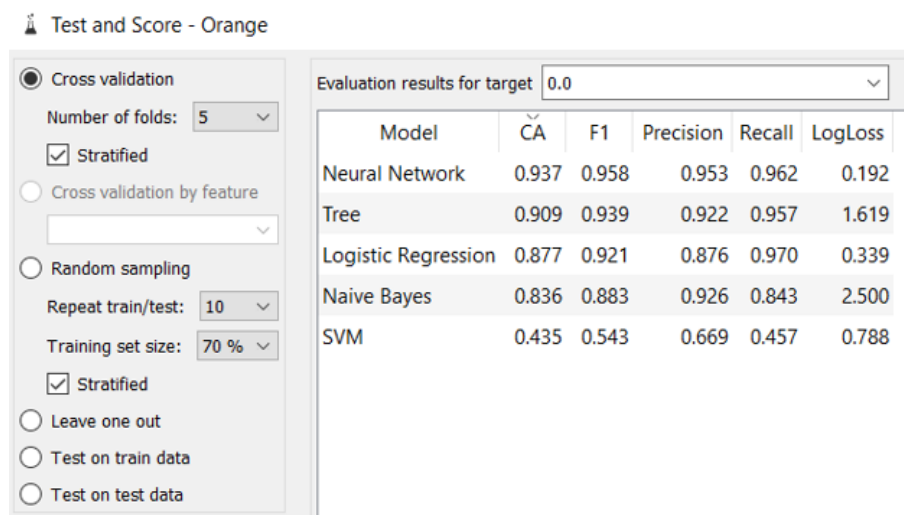


Figura B.3: Resultados da aplicação de vários algoritmos aos dados para a classe dos negativos.



# Bibliografia

- [Adobe, 2013] Adobe, U. (2013). Adobe audition.
- [Alhamid, 2020] Alhamid, M. (2020). What is cross-validation? <https://towardsdatascience.com/what-is-cross-validation-60c01f9d9e75>.
- [Badr, 2019] Badr, W. (2019). Having an imbalanced dataset? here is how you can fix it. *Towards Data Science*, 22.
- [Baptista, 2019] Baptista, B. (2019). Machine learning sem código. <https://medium.com/ensina-ai/machine-learning-sem-c%C3%B3digo-636d1a8f9081>.
- [Baughman et al., 2019] Baughman, A., Morales, E., Reiss, G., Greco, N., Hammer, S., e Wang, S. (2019). Detection of tennis events from acoustic data. In *Proceedings Proceedings of the 2nd International Workshop on Multimedia Content Analysis in Sports*, p. 91–99.
- [Brown, 2019] Brown, G. (2019). Digital audio basics: sample rate and bit depth. *Dostupno na: <https://www.izotope.com/en/learn/digital-audio-basics-sample-rate-and-bit-depth.html> [15. rujna 2020.]*.
- [Brownlee, 2020] Brownlee, J. (2020). Tour of evaluation metrics for imbalanced classification. *Vermont Victoria*.
- [Brownlee, 2021] Brownlee, J. (2021). How to choose an activation function for deep learning. *Machine Learning Mastery*.
- [Brownlee, 2022] Brownlee, J. (2022). Types of classification tasks in machine learning. *Machine Learning Mastery*, 4p. Available online at: <https://machinelearningmastery.com/types-of-classification-in-machine-learning> (accessed November 25, 2021).

- [Council et al., 2004] Council, N. R. et al. (2004). Hearing loss: Determining eligibility for social security benefits.
- [Courel-Ibáñez et al., 2019] Courel-Ibáñez, J., Martinez, B. J. S.-A., e Marín, D. M. (2019). Exploring game dynamics in padel: Implications for assessment and training. *The Journal of Strength & Conditioning Research*, 33(7):1971–1977.
- [Doshi, 2019] Doshi, S. (2019). Various optimization algorithms for training neural network. *Towards data science*, 13.
- [Evia, 2022] Evia, A. (visitado em Jul.2022). Nyquist sampling theorem.
- [Ganser et al., 2021] Ganser, A., Hollaus, B., e Stabinger, S. (2021). Classification of tennis shots with a neural network approach. *Sensors*, 21(17):5703.
- [InterviewArea, 2022] InterviewArea (visitado em Ago, 2022). How do i choose a batch size? <https://www.interviewarea.com/frequently-asked-questions/how-do-i-choose-a-batch-size>.
- [Janiesch et al., 2021] Janiesch, C., Zschech, P., e Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, 31(3):685–695.
- [Keras, 2022] Keras (visitado em Ago, 2022). About keras. <https://keras.io/about/>.
- [Kim e Pardo, 2018] Kim, B. e Pardo, B. (2018). A human-in-the-loop system for sound event detection and annotation. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 8(2):1–23.
- [Krishnan, 2021] Krishnan, S. (2021). How to determine the number of layers and neurons in the hidden layer. *URL: https://medium.com/geekculture/introduction-to-neuralnetwork-2f8b8221fbd3*.
- [LibreTexts, 2022a] LibreTexts, E. (2022a). Signal sampling. [https://eng.libretexts.org/Bookshelves/Electrical\\_Engineering/Signal\\_Processing\\_and\\_Modeling/Signals\\_and\\_Systems\\_\(Baraniuk\\_et\\_al.\\_\)/10%3A\\_Sampling\\_and\\_Reconstruction/10.01%3A\\_Signal\\_Sampling](https://eng.libretexts.org/Bookshelves/Electrical_Engineering/Signal_Processing_and_Modeling/Signals_and_Systems_(Baraniuk_et_al._)/10%3A_Sampling_and_Reconstruction/10.01%3A_Signal_Sampling).

- [LibreTexts, 2022b] LibreTexts, E. (2022b). Signal sampling. [https://eng.libretexts.org/Bookshelves/Electrical\\_Engineering/Signal\\_Processing\\_and\\_Modeling/Signals\\_and\\_Systems\\_\(Baraniuk\\_et\\_al.\)/10%3A\\_Sampling\\_and\\_Reconstruction/10.03%3A\\_Signal\\_Reconstruction](https://eng.libretexts.org/Bookshelves/Electrical_Engineering/Signal_Processing_and_Modeling/Signals_and_Systems_(Baraniuk_et_al.)/10%3A_Sampling_and_Reconstruction/10.03%3A_Signal_Reconstruction).
- [Liu et al., 2006] Liu, S., Xu, M., Yi, H., Chia, L.-T., e Rajan, D. (2006). Multimodal semantic analysis and annotation for basketball video. *EU-RASIP Journal on Advances in Signal Processing*, 2006:1–13.
- [MachineLearning, 2017] MachineLearning (2017). Harmonic precision-recall mean (f1 score). <https://machinelearning.wtf/terms/harmonic-precision-recall-mean-f1-score/>.
- [McFee et al., 2015] McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., e Nieto, O. (2015). librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, p. 18–25. Citeseer.
- [McNealis, 2020] McNealis, N. (2020). A simple introduction to dropout regularization (with code!). <https://medium.com/analytics-vidhya/a-simple-introduction-to-dropout-regularization-with-code-5279489dda1e>.
- [Medium, 2019] Medium (2019). A beginner intro to neural networks. <https://purnasaigudikandula.medium.com/a-beginner-intro-to-neural-networks-543267bda3c8>.
- [Meyda, 2022] Meyda (visitado em Jul.2022). Audio feature extraction for javascript. <https://meyda.js.org/audio-features.html>.
- [Muralidhar, 2021] Muralidhar, K. (2021). What is stratified cross-validation in machine learning? *Medium*. Retrieved December, 12:2021.
- [Nyuytiymbiy, 2021] Nyuytiymbiy, K. (2021). Parameters and hyperparameters in machine learning and deep learning. *Medium*. <https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac> (April 22, 2021).
- [Pierre, 2021] Pierre, S. (2021). A guide to regularization in python. <https://builtin.com/data-science/overfitting-regularization-python>.

- [Rauber, 2022] Rauber, T. W. (visitado em Ago, 2022). Redes neuronais artificiais. *Documento de Apoio (visitado em Jul, 2022)*.
- [Room, 2021] Room, C. (2021). Audio feature extraction. *machine learning*, 16(17):51.
- [Rosão, 2012] Rosão, C. M. T. (2012). *Onset detection in music signals*. PhD thesis.
- [Sah, 2020] Sah, S. (2020). Machine learning: a review of learning types.
- [Sampaio et al., 2006] Sampaio, R., Cataldo, E., e BRANDÃO, A. (2006). Análise e processamento de sinais. *Sociedade Brasileira de Matemática Aplicada e Computacional. São Paulo*.
- [Semmlow, 2012] Semmlow, J. (2012). Chapter 3 - fourier transform: Introduction. In Semmlow, J., editor, *Signals and Systems for Bioengineers (Second Edition)*, Biomedical Engineering, p. 81–129. Academic Press, second edition edition.
- [Tennisnerd, 2015] Tennisnerd (2015). What is padel tennis? <https://tennisnerd.net/padel/what-is-padel-tennis/15893>.
- [W3schools, 2022] W3schools (visitado em Set, 2022). What is ajax? [https://www.w3schools.com/whatis/whatis\\_ajax.asp](https://www.w3schools.com/whatis/whatis_ajax.asp).