NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE INFORMÁTICA

MIGUEL TEODORO MOREIRA

Bachelor Degree in Computer Science and Engineering

APRENDIZAGEM AUTOMÁTICA PARA TREINADOR VIRTUAL DE PADEL

MESTRADO EM ENGENHARIA INFORMÁTICA

Universidade NOVA de Lisboa
Fevereiro, 2022

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE INFORMÁTICA

APRENDIZAGEM AUTOMÁTICA PARA TREINADOR VIRTUAL DE PADEL

MIGUEL TEODORO MOREIRA

Bachelor Degree in Computer Science and Engineering

Orientador: Bruno Moscão
Diretor Técnico, SkillsWorkflow

Coorientadora: Maria Cecília Gomes
Professora Auxiliar, Universidade NOVA de Lisboa

Júri:

Presidente: Doutor Miguel Carlos Pacheco Afonso Goulão
Professor Associado, Universidade NOVA de Lisboa

Arguente: Doutor João Miguel da Costa Magalhães
Professor Associado, Universidade NOVA de Lisboa

Orientador: Eng. Bruno Miguel Condeça Moscão
Diretor Técnico, Skills Workflow

MESTRADO EM ENGENHARIA INFORMÁTICA

Universidade NOVA de Lisboa
Fevereiro, 2022

Aprendizagem Automática para Treinador Virtual de Padel

Agradecimentos

Neste capítulo deixo os mais sinceros agradecimentos a todos aqueles que tornaram a realização desta tese possível num ano tão complicado como o de 2021.

Quero agradecer em especial ao meu orientador Bruno Moscão, e a toda a equipa da SkillsWorflow, em especial para o João, Fernando e Hélder, pela oportunidade que me foi dada de realizar esta dissertação, integrada no contexto da empresa e do seu trabalho.

À minha co-orientadora, Prof. Maria Cecília Gomes, quero agradecer pelos conselhos, críticas construtivas e pelo forte apoio durante esta dissertação. Ao Prof. Ludwig Krippahl também agradecer os conselhos dados.

A todos os amigos de longa data, aos amigos que fiz durante o meu percurso universitário e na minha aventura de Erasmus, e em especial àqueles que partilharam comigo esta jornada, obrigado a todos.

Por último, mas não menos importante, deixo um enorme agradecimento à minha família, especialmente aos meus pais e irmãs, por todo o apoio, compreensão e paciência que tiveram para comigo ao longo do meu percurso académico.

Resumo

A análise desportiva é uma área de investigação emergente com aplicações em temas como a previsão do desempenho de equipas, estimativa do valor de mercado de jogadores e previsão de possíveis lesões. Jogadores e treinadores estão dispostos a integrar ferramentas capazes de oferecer recomendações e melhorar o desempenho dos jogadores.

A indústria do desporto gera grandes conjuntos de dados. Os dados gerados têm enorme potencial e costumam estar representados em formato vídeo. A grande disponibilidade de dados promove a aplicação de áreas como a Inteligência Artificial, Aprendizagem Automática e Visão por Computador.

O Padel é um jogo de raquetes que tem ganho participantes em todo o mundo, tendo sido considerado recentemente como o desporto com maior taxa de crescimento. Sem sinais de abrandamento a profissionalização de competições e atletas é crescente. Os grandes desportos de raquetes como o ténis, ténis de mesa ou o badmínton já têm ferramentas capazes de extrair dados dos vídeos e recomendar algumas estratégias. Como desporto diferente e emergente, o Padel tem uma componente de estratégia por explorar.

Esta dissertação propõe uma ferramenta que recebe um vídeo de uma sessão de Padel e recomende uma melhor estratégia de jogo ao jogador do que as que os jogadores apresentaram na sequência de vídeo. A ferramenta é dividida em duas grandes componentes: Sistema de Visão e Treinador Virtual.

O Sistema de Visão tem como função detetar e seguir individualmente cada jogador recorrendo a algoritmos de deteção e *tracking* de objetos como o YOLOv4 e Deep SORT. A componente de Treinador Virtual aplica um modelo de Aprendizagem Automática sobre os dados referentes à posição de cada jogador e sugere ao jogador uma melhor estratégia em cada jogada efetuada. O modelo de Aprendizagem mencionado foi treinado com um conjunto de dados recolhido e posteriormente anotado por dois jogadores experientes de Padel.

Palavras-chave: Padel, Visão por Computador, Aprendizagem Automática, Análise Desportiva

Abstract

Sports analytics is an emerging area of research with applications in topics such as predicting team performance, estimating the market value of players, and predicting potential injuries. Players and coaches are willing to integrate tools that can provide recommendations and improve player performance.

The sports industry generates large data sets. The data generated has enormous potential and is usually represented in video format. The large availability of data promotes the application of areas such as Artificial Intelligence, Machine Learning and Computer Vision.

Padel is a game of rackets that has been gaining participants all over the world, and was recently considered the sport with the highest growth rate. With no signs of slowing down, the professionalization of competitions and athletes is growing. The big racket sports like tennis, table tennis or badminton already have tools capable of extracting data from videos and recommending some strategies. As a different and emerging sport, Padel has an unexplored strategy component.

This dissertation proposes a tool that receives a video of a Padel session and recommends a better game strategy to the player than the ones the players presented in the video sequence. The tool is divided into two major components: Vision System and Virtual Coach.

The Vision System has the function of detecting and tracking each player individually using object detection and tracking algorithms such as YOLOv4 and Deep SORT. The Virtual Trainer component applies a Machine Learning model on player's position data and suggests a better strategy to the player for each move. The learning model mentioned above was trained with a data set collected and later annotated by two experienced padel players.

Keywords: Padel, Computer Vision, Machine Learning, Sports analytics

Índice

Índice de Figuras

Índice de Tabelas

Siglas

AA Aprendizagem Automática 6, 25, 38, 42, 43, 45, 46, 62, 63
AD Árvore de Decisão x, xi, 8, 42, 44, 47, 48, 49, 53, 55, 56, 58, 59
AUC Area Under The Curve 14, 15, 53

CNN *Convulotional Neural Network* 16, 19, 29

FA Florestas Aleatórias x, xi, 10, 42, 44, 47, 48, 50, 53, 54, 55, 56, 57, 58, 60, 63
FN Falso Negativo 13, 14, 20, 21, 22, 31
FP Falso Positivo 13, 14, 15, 20, 22, 31

KNN K vizinhos mais próximos x, xi, 10, 42, 44, 45, 47, 48, 51, 53, 54, 55, 57, 58, 60

mAP *Mean Average Precision* 21, 62
MOT *Multi-Object Tracking* 21, 23, 29

ROC Receiver Operating Characteristics 14, 15, 53

SVM Support Vector Machine x, xi, 7, 8, 42, 44, 47, 48, 52, 53, 55, 56, 58, 59, 61

VC Validação Cruzada 42, 45, 47
VN Verdadeiro Negativo 13, 15
VP Verdadeiro Positivo 13, 14, 20, 21, 22, 31

1

Introdução

1.1 Introdução

O Padel é um jogo de raquetes que nasceu em 1969 no México, após Enrique Corcuera adaptar o seu campo caseiro de Squash com elementos do ténis. O desporto foi ganhando participantes por toda a América latina e, em simultâneo, em Espanha o Padel vai ganhando popularidade até que em 2014 ultrapassa o ténis em número de atletas com 6 milhões de jogadores e 5500 campos [41]. O Padel foi considerado o desporto com a maior taxa de crescimento no mundo e não mostra sinais de abrandar. Em 2013 foi realizado o primeiro circuito mundial de Padel, o qual tem sido repetido todos os anos, permitindo o aumento do número de jogadores profissionais [44].

As razões que suportam o crescimento do desporto são a simplicidade e facilidade de aprendizagem. Atletas que já tenham jogado outros desportos com raquetes como ténis ou badmínton têm facilidade de adaptação. Uma das razões que sustenta a popularidade do desporto é o aspeto social, dado que o jogo é disputado por duplas num campo menor que o do ténis, o que permite aos atletas estarem mais próximos e em constante comunicação. Outra razão deve-se ao lado financeiro, os campos de Padel são consideravelmente menores o que motiva a alguns proprietários a transformar os antigos campos de ténis em dois campos de Padel. Os preços do alugar um campo são também menores, pois são divididos por quatro jogadores [42].

Durante os jogos e sessões de treino, os praticantes de Padel sentem a necessidade de receber feedback sobre o seu posicionamento e jogadas efetuadas. As soluções existentes são orientadas para as grandes competições de ténis, por permitirem o investimento em toda uma infraestrutura especializada. A tecnologia mais popular é o Hawk-Eye [31] que necessita da montagem de seis câmaras profissionais de alto desempenho. Este tipo de tecnologia não é aplicável nos frequentes campos de Padel utilizados por jogadores amadores, devido às piores infraestruturas físicas e ao elevado custo dos equipamentos.

Ao mesmo tempo, a análise desportiva tem ganho muito interesse tanto para treinadores como para atletas que pretendem ter os melhores planos de treino e estratégias para melhorarem o seu desempenho. O estudo *Sports Analytics Market Size, Share, Trends*

Analysis [36] prevê o crescimento anual composto de 21.3% do mercado de análise de desporto entre 2017 e 2028 nos Estados Unidos. O mercado de análise foi avaliado em 885.0 milhões de dólares em 2020. A demanda nos próximos anos prevê-se ser tanto em serviços como em software.

O progresso da análise desportiva e a crescente popularidade do Padel oferecem uma oportunidade de negócio que a SkillsWorkflow decidiu explorar. O desenvolvimento de uma ferramenta capaz de extrair informação presente nos vídeos do Padel é uma mais valia tanto para jogadores como para treinadores.

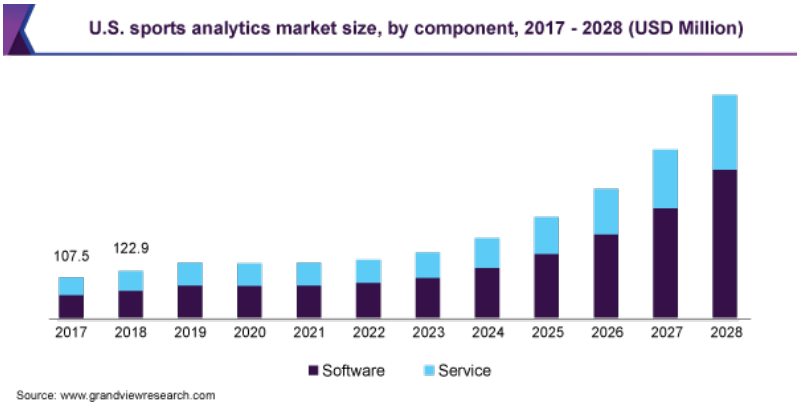


Figura 1.1: Análise do mercado de análises desportivas nos EUA

1.2 Definição do problema e objetivos do trabalho

Os desportos de raquetes mais populares como ténis, ténis de mesa e badmínton já têm algumas soluções que permitem extrair informações presentes nas gravações de jogos. No entanto, as soluções existentes não se adequam às características do Padel. O Padel tem características únicas que provocam novos problemas. As diferenças começam logo com o campo de Padel que difere dos demais e provoca novos problemas na deteção automática do campo. Os vídeos de Padel são gravados de uma posição alta e consequentemente apresentam a imagem numa certa perspetiva, o que resulta em que a metade do campo mais longe da câmara tenha dimensões inferiores em píxeis à metade mais próxima. O mapeamento do campo é necessário para resolver este problema.

O Padel é disputado em duplas, os jogadores durante o jogo estão em constante movimento e frequentemente sobrepostos ou parcialmente ocultados, assim sendo é importante detetar e atribuir uma identidade constante a cada jogador, de forma a saber as posições que ocupou durante o jogo. Os desportos de raquetes anteriormente enunciados dispensam o processo de monitorização individual visto em que cada metade do campo existe apenas um jogador.

Como em qualquer desporto, o Padel tem a sua parte de estratégia. Os jogadores com melhor conhecimento tático do jogo tendem a obter melhores desempenhos. A aprendizagem de estratégias por parte dos jogadores está dependente do contacto e dicas de

treinadores ou de jogadores mais experientes. O problema presente é ensinar aos jogadores se as jogadas que efetuaram numa sequência de vídeo serão as melhores e apresentar ao atleta uma melhor jogada possível.

Exposto o problema desta dissertação, são enunciados os seguintes objetivos:

- Mapear automaticamente o campo presente na imagem proveniente da gravação do jogo para um modelo de campo 2D.
- Detetar e monitorizar os jogadores presentes no campo.
- Sugerir aos jogadores uma melhor estratégia nos momentos em que jogam a bola.

1.3 Proposta de solução e contribuições esperadas

De forma a resolver os problemas enunciados e cumprir os objetivos, a solução proposta é o desenvolvimento de duas componentes. A primeira componente é a elaboração de um sistema de visão capaz de detetar e monitorizar cada jogador durante uma sequência de vídeo. A segunda componente é o desenvolvimento de um treinador virtual cujo objetivo é indicar a cada jogada qual o melhor local para onde bater a bola. Como jogada é entendido o momento em que o jogador bate com a raquete na bola.

O sistema de visão receberá como *input* sequências de vídeo captadas por apenas uma câmara. A escolha de apenas uma câmara é justificada pela facilidade de montagem, baixos custos associados e disponibilidade de vídeos existentes com esta característica nas plataformas de vídeo.

As componentes vão ser desenvolvidas independentemente uma da outra e testadas em conjunto na última etapa do projeto. O grande foco desta dissertação será o treinador virtual. O desenvolvimento do sistema de visão é indispensável, visto que através desta componente serão extraídas informações a ter em conta na escolha da melhor jogada pelo treinador virtual.

As contribuições esperadas decorrem do desenvolvimento das duas componentes. Na primeira componente, sistema de visão, é esperado o treino de um detetor de objetos especializado para a deteção de jogadores de Padel e a aplicação de um algoritmo de *tracking*.

Na segunda componente, treinador virtual, as contribuições esperadas são:

- Construção de um conjunto de dados com jogadas ideais em diversas situações de jogo.
- Desenvolvimento de um classificador capaz de apontar a jogada ideal ao jogador nas situações que se encontra durante o jogo.

Os dados gerados pelo Sistema de Visão e Treinador Virtual serão fonte de dados para uma aplicação móvel e web dedicada ao Padel.

1.4 Estrutura do documento

O resto deste documento de preparação da dissertação está estruturado da seguinte maneira:

- O capítulo 2 começa por descrever sucintamente as regras e sistema de pontos do Padel. De seguida são abordados conceitos chave de aprendizagem automática e visão por computador. Por fim são descritos trabalhos relacionados na área de análise desportiva suscetíveis de serem aplicados no nosso problema.
- O capítulo 3 descreve um pequeno sistema de visão capaz de detetar e monitorizar jogadores de Padel num vídeo de uma partida. O sistema é acompanhado com representação dos dados extraídos num modelo do campo 2D.
- O capítulo 4 descreve e explica as metodologias e escolhas optadas no desenvolvimento do treinador virtual.
- O capítulo 5 apresenta os resultados obtidos da solução desenvolvida no capítulo anterior.
- Por fim, o capítulo 6 contém as conclusões da solução obtida e propostas de trabalho futuro.

2

Trabalho relacionado

2.1 Padel

O jogo de Padel é um jogo de raquetes que envolve a competição entre duas duplas. O jogo é disputado no campo com as dimensões padrão, a figura 2.1 mostra as dimensões associadas. Nesta secção vai ser apresentado um apanhado dos fundamentos do jogo e sistema de pontos, para informações mais detalhadas sobre as regras, consultar a informação disponibilizada pela federação portuguesa de Padel [12].

O jogo tem uma estrutura hierárquica em que ganhar pontos leva a ganhar jogo, ganhar jogos leva a ganhar o set e por fim ganhar sets leva a ganhar a partida. Os pontos são iniciados com o serviço, este para ser válido, o jogador ao servir tem de começar por deixar bater a bola no chão atrás da linha de serviço e depois bater na bola de forma a passar a rede com uma trajetória diagonal e acertar na zona de receção do campo adversário. Se o serviço for válido o jogo continua com as duplas a baterem a bola para o campo adversário. Cada dupla tem que jogar a bola antes que toque pela segunda vez no seu campo, em caso de insucesso o ponto termina e é atribuído à dupla adversária. A bola durante os pontos pode ressaltar na parede, sendo este momento que diferencia o Padel dos restantes jogos de raquetes. O sistema de pontos é semelhante ao ténis em que cada dupla ganha pontos com a seguinte ordem: zero pontos - 0, primeiro ponto - 15, segundo ponto - 30, terceiro ponto - 40, quarto ponto ganhou o jogo. O par que ganhar seis jogos com a diferença de dois para a outra dupla ganha o set, caso contrário joga-se o "tie-break"para desempatar o set. As partidas de Padel são jogadas à melhor de três sets. No "tie-break"os pontos são contados um, dois, três, quatro e daí adiante até sete, quem chegar primeiro aos sete pontos ganha desde que o faça com uma vantagem de dois pontos, caso contrário o "tie-break"continua até que a esta margem seja conseguida.

O ténis é um dos desportos mais semelhantes ao Padel. As transmissões televisivas de ambos os desportos são semelhantes. Em ambas 5.8 é possível ver os campos na sua totalidade, o que permite saber sempre a posição do jogador, evitando assim lidar com problemas de oclusão total do jogador. Em ambos os campos existe uma rede a separar

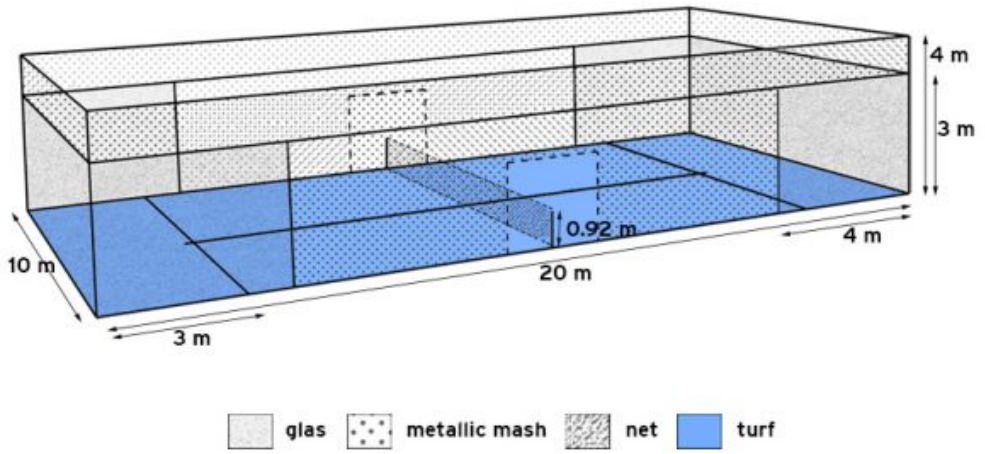
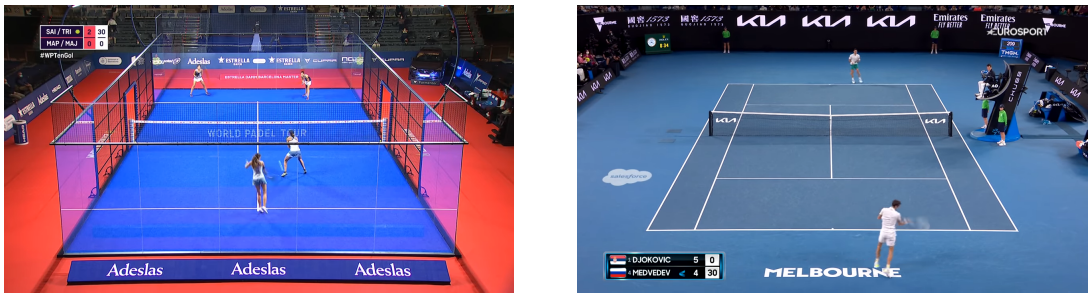


Figura 2.1: Campo de Padel [41]

cada equipa e por fim o tamanho dos jogadores em pixeis é semelhante.

Devido às semelhanças entre ambos os desportos, o trabalho de pesquisa foi focado no ténis, focando nas técnicas de Processamento de Imagem, Aprendizagem Automática e Visão por Computador que possam ser aplicadas no Padel.



(a) World Padel Tour (b) Australia Open

Figura 2.2: Transmissões televisivas de padel e ténis [44] [1]

2.2 Aprendizagem Automática Supervisionada

Aprendizagem Automática (AA) é definido como o paradigma computacional, onde a capacidade de resolver um problema é desenvolvida através de exemplos passados. No contexto de AA, os exemplos passados que são usados para construir a capacidade são chamados dados de treino e o processo de capacitação é a aprendizagem.

A Aprendizagem Supervisionada é o tipo de aprendizagem, em que a capacidade de aprendizagem do algoritmo é construída sobre dados rotulados. Aos algoritmos são fornecidos dados de treino com vários atributos e um rótulo ou classe. A aprendizagem supervisionada é normalmente aplicada a problemas de regressão e classificação.

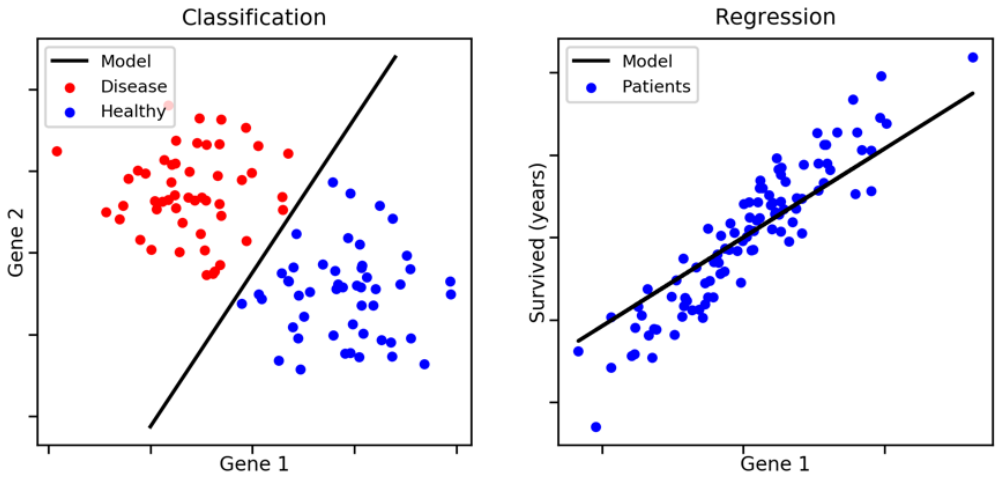


Figura 2.3: Classificação e Regressão

2.2.1 Classificação

A classificação é definida como o problema de atribuir uma classe entre um conjunto predefinido a uma amostra de dados. A cada amostra é associado um valor qualitativo correspondente a uma classe. Na prática, a classificação consiste no processo de encontrar uma função que melhor define uma fronteira entre pontos que representam classes diferentes. Por exemplo, na figura 2.3, o cenário é tendo em conta dois genes determinar o paciente como saudável ou doente.

2.2.2 Regressão

A regressão é definida como o problema de estimar o valor contínuo a uma amostra de dados. A cada amostra é associado um valor quantitativo. Na prática, a regressão consiste no processo de encontrar uma linha ou curva que generalize bem a maioria dos pontos pertencentes aos dados de treino. Por exemplo, na figura 2.3, o cenário é tendo em conta um determinado gene determinar o número de anos de vida de um paciente.

2.2.3 Algoritmos de Classificação

Apresentação e descrição de algoritmos frequentemente aplicados em Aprendizagem Automática Supervisionada.

2.2.3.1 *Support Vector Machine*

Support Vector Machine (SVM) é um algoritmo de aprendizagem supervisionada muito versátil, capaz de ser usado tanto em problemas de regressão como de classificação. O algoritmo é especialmente adequado para conjuntos de dados de pequena ou média dimensão já que segundo [13] apenas um quarto da quantidade recomendada de dados é necessária para produzir bons resultados.

Em problemas de classificação, o algoritmo SVM tem como objetivo encontrar o hiperplano num espaço com dimensão N, sendo o N o número de características, que melhor divida as amostras de classes diferentes. Os hiperplanos são vistos como divisores entre classes. Entre amostras de duas classes é possível que existam vários hiperplanos possíveis, por isso o algoritmo tem como objetivo maximizar a margem entre os dados de treino e o hiperplano. Uma maior distância de margem permite que as futuras amostras sejam classificadas com maior exatidão. Para alcançar a máxima distância possível são usados os *support vectors* que são pontos perto do hiperplano que influenciam a posição e orientação do hiperplano com intuito de maximizar a margem do classificador. Na figura 2.4 está ilustrado uma visão geral dos conceitos.

O classificador SVM tem dois parâmetros que podem ser ajustados merecedores de discussão. O primeiro é representado pela letra C, que controla o balanço entre a margem de decisão mais permeável e a exatidão da classificação das amostras. O C com um valor mais alto, tende a um maior número de classificações corretas, devido ao algoritmo ter uma margem de decisão menos permeável e consecutivamente menor tolerância a classificações incorretas. Contudo, um valor de C mais alto pode levar a problemas de sobre ajustamento e piores resultados com dados nunca antes vistos. O C com um valor menor implica que as margens de decisão serão mais permissivas e gera um maior número de classificações incorretas. Porém, o classificador vai ser genericamente melhor e poderá ter melhores resultados com dados nunca antes vistos.

O outro parâmetro é a função de *kernel*. Uma função matemática de *kernel* recebe os dados de entrada e transforma-os num espaço dimensional mais elevado, onde é possível a classificação. Entre as funções de *kernel* possíveis estão a linear, polinomial e a mais aplicada *Gaussian Radial Basis*.

A principal desvantagem deste modelo é que a sua aplicação é normalmente uma *black box*, isto é, o conhecimento apreendido durante o treino não é interpretável, só se conhece os dados de entrada e saída, o que torna possíveis melhorias de desempenho complicadas.

2.2.3.2 Árvores de Decisão

As Árvore de Decisão (AD) são um modelo hierárquico de aprendizagem supervisionada, onde os dados são continuamente divididos de acordo com certos parâmetros. A árvore de decisão é composta por raiz, nós de decisão e folhas. Cada nó de decisão implementa uma função de teste que ajuda a discriminar o resultado final da classificação ou regressão. Com os dados de entrada, em cada nó, é efetuado um teste e com o resultado do mesmo é tomado um ramo. O processo começa na raiz ou base e é repetido recursivamente até chegar a uma folha. O valor da folha corresponde ao resultado final.

A árvore de decisão é um modelo não paramétrico no sentido em que não é assumida nenhuma forma de parâmetro para a densidade das classes nem para a estrutura da árvore. A estrutura da árvore depende da complexidade do problema e conjunto de dados, sendo

que a árvore cresce a adicionar novos ramos e folhas.

A estrutura hierárquica da árvore permite uma rápida computação do resultado final já que, por exemplo numa árvore binária, a cada decisão e no melhor caso, metade dos possíveis casos são eliminados. Outra vantagem é as árvores podem ser convertidas num conjunto de regras IF-THEN que são facilmente interpretáveis. Por esta razão, as árvores são um algoritmo popular e por vezes preferido em relação a outros com maior exatidão.

Uma desvantagem deste tipo de algoritmos é estarem sujeitos a sobre ajustamento, quando a árvore gerada se torna demasiado grande e complexa. Uma técnica para reduzir este problema é o pruning ou em português a poda, em que sub-árvores em certos níveis são alteradas de forma a reduzir os erros nas classificações efetuadas.

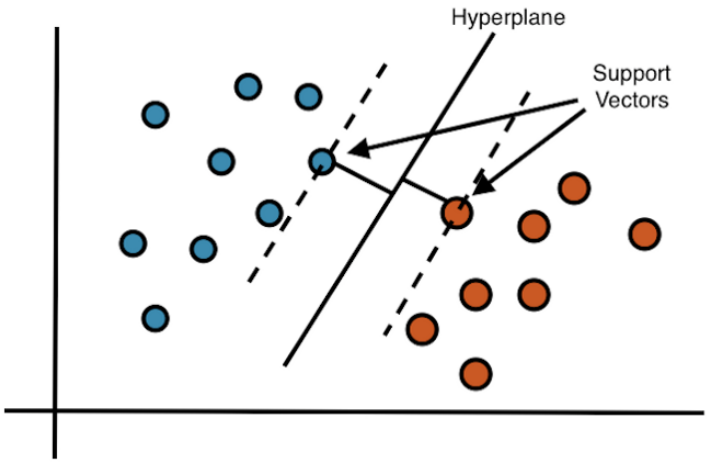


Figura 2.4: *Support Vector Machine*

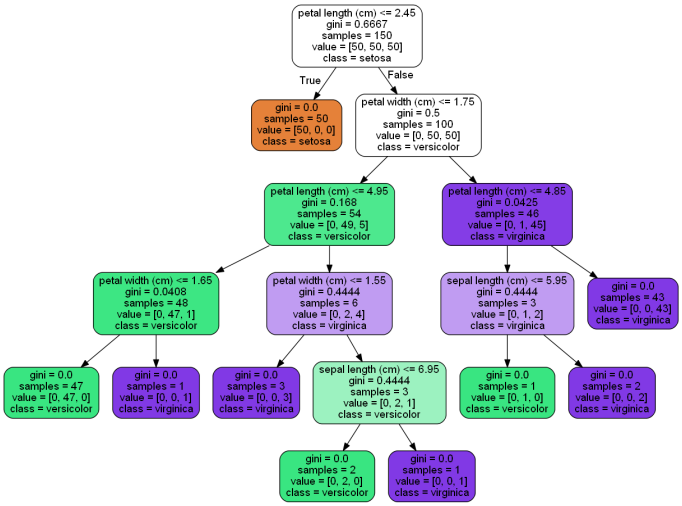


Figura 2.5: Exemplo Árvore de Decisão

2.2.3.3 Florestas Aleatórias

Florestas Aleatórias (FA) é um algoritmo de aprendizagem supervisionada e pertence ao grupo de algoritmos de aprendizagem combinada, na literatura *ensemble learning*. Este tipo de aprendizagem combina vários modelos base de forma a agregar todas as previsões e criar um modelo mais exato. Existem alguns tipos de aprendizagem combinada, mas o usado no algoritmo FA é o *bagging*. No *bagging*, os vários modelos são treinados com parcelas de dados independentemente e no final o resultado é escolhido com base na maioria de votos dos modelos.

Tal como o nome sugere, o algoritmo FA é composto por várias árvores de decisão. Na fase de treino, o conjunto de dados de treino é dividido aleatoriamente em parcelas e são geradas as árvores para cada conjunto. Além da divisão dos dados de treino, é efetuada a partição de atributos, ou seja, cada árvore além de ver apenas uma parcela dos dados também só vê um subconjunto de atributos. Esta característica faz com que os estimadores ou árvores de decisão quando avaliados individualmente sejam essencialmente fracas, contudo quando consideradas em conjunto têm maior assertividade nas suas previsões. Na figura 2.6 apresenta um esquema geral do funcionamento das Florestas Aleatórias, focando na questão da partição dos atributos.

O algoritmo FA é tido em conta como uma melhor alternativa às árvores de decisão no problema do sobre ajustamento dos dados de treino, pois ao existirem várias árvores treinadas com parcelas do conjunto de dados, faz com que o algoritmo não se ajuste em demasia aos dados de treino. Contudo, por agregar um grande número de árvores de decisão as FA tem uma maior complexidade, o que implica maiores recursos computacionais e neste caso mais tempo de treino em comparação com outros algoritmos com os mesmos objetivos. Um hiperparâmetro ajustável é o número de estimadores, se houver um maior número de estimadores, pode ser que o desempenho do algoritmo melhore, mas o tempo de execução também tende a crescer. E o caso contrário para quando há um menor número de estimadores.

2.2.3.4 K vizinhos mais próximos

O K vizinhos mais próximos (KNN) é um algoritmo de aprendizagem supervisionada e tem um método de aprendizagem preguiçosa, no qual a generalização dos dados de treino é apenas realizada quando é pedida uma previsão ao modelo. O KNN assume que dados semelhantes têm os mesmos resultados. Para computar a semelhança é utilizada uma função de distância entre pontos.

O funcionamento do algoritmo KNN pode ser dividido nos seguintes passos:

- Os dados de treino são carregados e é definido o k, número de vizinhos, i.e. quantos pontos são considerados para o resultado final.
- Calcula-se a distância dos dados de treino aos dados de teste através de uma função de distância entre pontos. De seguida são ordenadas as distâncias e guardadas as k

entradas mais próximas.

- As k entradas mais próximas dos dados de teste são usadas para o resultado final, se o problema for de regressão, o resultado é a média das k entradas, caso o problema seja de classificação, o resultado é a classe da maioria das k entradas.

Os dados devem possuir atributos numéricos para ser possível calcular a distância entre pontos. Entre as distâncias normalmente usadas estão a distância Euclidiana, Manhattan e Minkowski.

A desvantagem deste modelo é que em problemas com muitos dados de treino, as exigências computacionais são maiores, por necessitar de memória para guardar os dados de treino que depois são usadas para computar as distâncias com as amostras, tornando a previsão do modelo tendencialmente mais lenta.

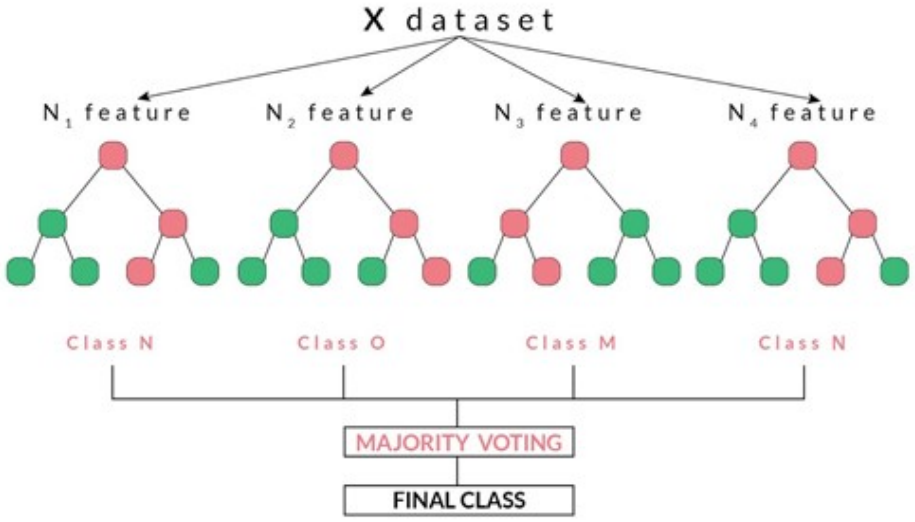


Figura 2.6: Estrutura das Florestas Aleatórias

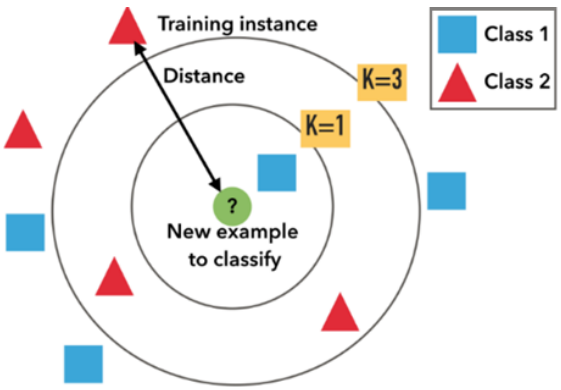


Figura 2.7: Esquema do algoritmo K vizinhos mais próximos

2.2.3.5 Redes neuronais

Redes neuronais artificiais são sistemas de computação que tentam simular o processo de decisão nas redes neuronais biológicas. As redes neuronais artificias tais como as biológicas são compostas por unidades de processamentos denominadas por neurónios. A rede neuronal mais simples é composta por apenas um neurónio:

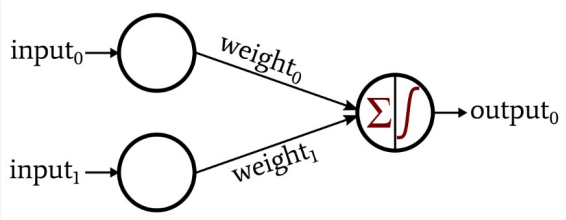


Figura 2.8: Neurónio [22]

Com a simples rede neuronal mencionada acima não se pode fazer muita coisa, mas ao estruturar e combinar diferentes neurónios é formada uma rede neuronal mais complexa designada por rede neuronal multi camadas. A rede neuronal multi camadas possui uma camada de input, um número variável de camadas escondidas e por fim uma camada de output. Cada camada recebe como input o *output* da camada anterior e por fim a última camada retorna o *output* final.

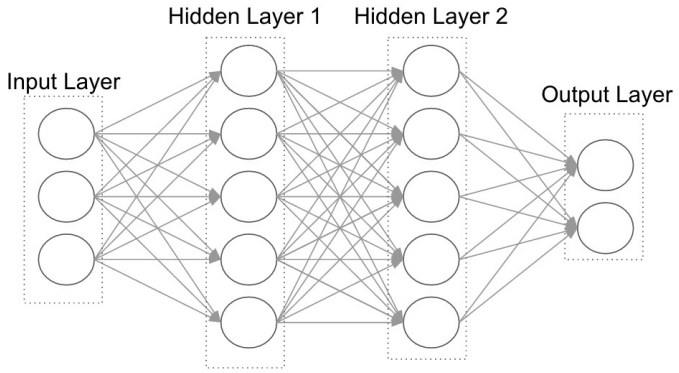


Figura 2.9: Rede neuronal multi camadas [29]

2.2.4 Métricas de Avaliação

Após o treino dos modelos, reunir um conjunto de métricas de avaliação é essencial para decidir o modelo que melhor resolve o problema. As escolhas das métricas que usamos na avaliação é fundamental na comparação de modelos.

2.2.4.1 Matriz de confusão

A matriz de confusão é uma das métricas mais usadas para avaliar o desempenho dos modelos em problemas de classificação. A matriz de confusão representa, numa tabela,

os 4 casos possíveis resultantes da classificação de um exemplo:

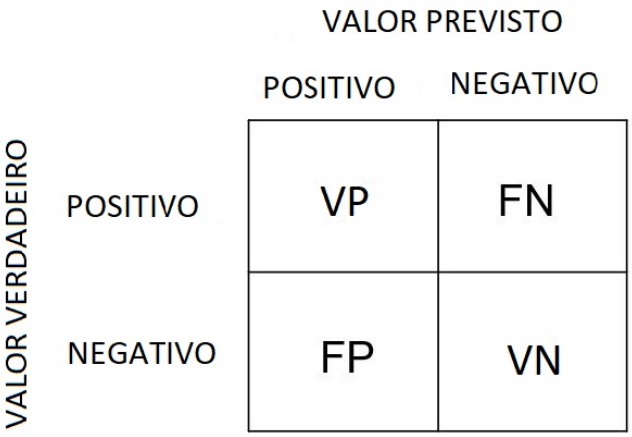


Figura 2.10: Matriz de confusão

- Verdadeiro Positivo (VP): a classificação do exemplo é positiva e a verdadeira classe é positiva.
- Verdadeiro Negativo (VN): a classificação do exemplo é negativa e a verdadeira classe é negativa.
- Falso Positivo (FP): a classificação do exemplo é positiva e a verdadeira classe é negativa.
- Falso Negativo (FN): a classificação do exemplo é negativa e a verdadeira classe é positiva.

O FP e FN representam os casos em que o modelo previu incorretamente a classe verdadeira do exemplo, os dados apresentados na tabela 2.10 ajudam no cálculo das métricas descritas nas próximas secções.

2.2.4.2 Exatidão

A Exatidão é a métrica mais comum em problemas de classificação e indica a percentagem de previsões corretas efetuadas pelo modelo sobre todas as previsões. A exatidão é uma métrica a aplicar quando o conjunto de dados tem a percentagem de classes equilibrada.

$$\textbf{Exatidão} = \frac{VP + VN}{VP + VN + FP + FN} \tag{2.1}$$

2.2.4.3 Relatório de Classificação

O Relatório de Classificação inclui um conjunto de métricas usadas para medir a qualidade das previsões. O relatório inclui as seguintes métricas:

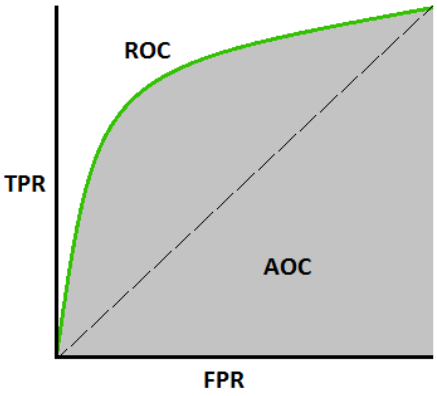


Figura 2.11: Curva AUC-ROC

- Precisão indica a percentagem de exemplos corretamente identificados, dentro daqueles que foram previstos pertencer à classe positiva. Uma interpretação comum é a capacidade do classificador não rotular um exemplo positivo que, na verdade, é negativo.

$$\textbf{Precisão} = \frac{VP}{VP + FP} \tag{2.2}$$

- Abrangência indica a percentagem de exemplos identificados como pertencendo à classe positiva, de entre todos os exemplos cuja verdadeira classe é positivas. Uma interpretação comum é a capacidade do classificador encontrar todas as instâncias positivas.

$$\textbf{Abragência} = \frac{VP}{VP + FN} \tag{2.3}$$

- F1-score é a média harmónica ponderada da precisão e abrangência.

$$\textbf{F1-score} = 2X\frac{\textbf{Precisão} \times \textbf{Abragência}}{\textbf{Precisão} + \textbf{Abragência}} \tag{2.4}$$

- *Support* indica o número de ocorrências de cada classe.
- *Macro Average* calcula a métrica independentemente de cada classe e, em seguida a média tratando todas as classes por igual.
- *Weighted Average* calcula a métrica com base no *support*.

2.2.4.4 Area Under The Curve (AUC) – Receiver Operating Characteristics (ROC)

A curva ROC é o gráfico que avalia o desempenho do modelo de classificação em função de dois parâmetros, a Taxa de Verdadeiros Positivos ou Abrangência no eixo y e a Taxa de

Falsos Positivos no eixo x.

$$\text{Taxa de Falsos Positivos} = \frac{FP}{FP + VN} \tag{2.5}$$

A AUC mede a área bidimensional abaixo da curva ROC e representa o grau de separabilidade entre duas classes. O valor da AUC varia entre 0 até 1, quanto maior o valor de AUC, melhor é a capacidade de efetuar previsões corretas. Um classificador com uma AUC igual a 0, indica que o classificador falhou todas as classificações enquanto um classificador com AUC igual a 1, indica que o classificador acertou todas. Um classificador com AUC igual a 0,5, indica que o classificador não consegue descriminar entre classes.

As curvas ROC são uma boa métrica de desempenho quando os dados não são desequilibrados e é tão importante os verdadeiros negativos como os verdadeiros positivos.

2.3 Visão por computador

Visão por computador é a área científica focada na replicação do complexo sistema de visão humana e em habilitar computadores a processar e entender imagens e vídeos da mesma forma que os humanos fazem. Avanços nas áreas de inteligência artificial e aprendizagem profunda, nomeadamente em redes neuronais permitiram o desenvolvimento acentuado desta área nos últimos anos.

A aprendizagem profunda mudou a abordagem que era até então feita em visão por computador, até à mudança os trabalhos de visão por computador eram realizados inicialmente por técnicas tradicionais que se focam na informação diretamente dada pelos píxeis para comparar imagens ou detetar objetos simples, através da cor, níveis de luminosidade, forma, etc. A aprendizagem profunda tem como base a utilização de redes neuronais que quando treinadas conseguem extrair padrões e produzir informação sobre os dados. Grandes conjuntos de imagens como ImageNet [38], CityScapes [9] e COCO [25] possuem milhões de imagens devidamente rotuladas e são fonte de dados para algoritmos de aprendizagem profunda.

Algumas subáreas da visão por computador são deteção de objetos, *tracking* de vídeo, estimativa de pose 3D, deteção de eventos, entre outras. Aplicações presentes no quotidiano são o reconhecimento facial, carros autónomos e sistemas de vídeo vigilância.

2.3.1 Deteção de objetos

A deteção de objetos é o processo em visão por computador de detetar certas classes de objetos numa imagem digital, exemplos de classes são pessoas, animais, carros, entre outros. O objetivo da deteção de objetos é detetar e localizar na imagem as instâncias de objetos pertencentes uma lista predefinida de classes. Assim para cada objeto detetado, o algoritmo de deteção estima a posição, dimensões e classe de cada objeto. A posição combinada com as dimensões do objeto forma a caixa circundante que delimita o objeto na imagem.

Os detetores de objetos são divididos em dois tipos, de uma fase e de duas fases. Os detetores de duas fases começam por gerar possíveis caixas circundantes, dividindo a imagem em regiões de interesse, estas regiões são separadas e posteriormente classificadas por uma *Convulotional Neural Network* (CNN). R-CNN [15] e Fast R-CNN [14] são exemplos de detetores de duas fases. Ao contrário dos detetores anteriores, o detetor de uma fase estima as caixas circundantes e a classe do objeto em apenas uma passagem sobre a rede CNN. YOLO [35] e SSD [26] são exemplos de detetores de uma fase. Normalmente, os detetores de duas fases apresentam uma maior taxa de exatidão em comparação com os de uma fase, mas à custa de uma menor velocidade de execução.

Um conceito presente nos algoritmos de deteção de objetos é o IoU, figura 2.12, que descreve a sobreposição da caixa circundante prevista com caixa circundante verdadeira. O quociente é dado pela área de sobreposição sobre a área de união de duas caixas circundantes. O IoU varia entre 0 e 1, quanto maior o valor maior a sobreposição das caixas.

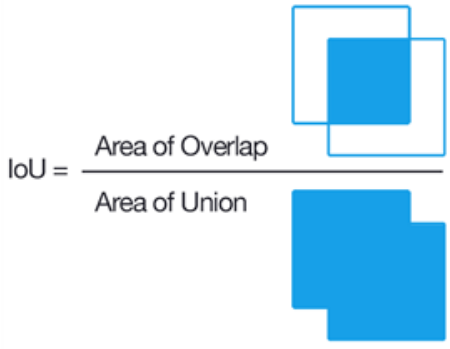


Figura 2.12: IoU

2.3.1.1 YOLO

Em 2015, Redmon et al. [35] introduziram uma abordagem inovadora à deteção de objetos a que chamaram YOLO, *You Only Look Once*.

No algoritmo YOLO, a imagem recebida é dividida numa grelha com dimensões iguais SxS. Cada célula da grelha é respetivamente responsável por prever as caixas circundantes presentes no seu espaço e uma taxa de confiança relativa à presença do objeto na caixa. A taxa de confiança é calculada como P (Objeto) * IoU, onde P(objeto) é a probabilidade de a caixa circundante conter o objeto e o IoU é a estimada intercessão sobre a união (IoU) entre a caixa prevista e a caixa verdadeira. Em cada célula são também calculadas as probabilidades de cada classe de objetos e com as caixas circundantes previstas formam uma pontuação combinada. As pontuações mais altas são consideradas as deteções finais do algoritmo.

O processo descrito é realizado por uma rede neuronal de arquitetura CNN inspirada no GoogleLeNet [40]. A arquitetura do algoritmo YOLO consiste em 24 camadas de convolução com 2 camadas *fully connected* no final. Para cada caixa circundante são definidos

5 valores, x, y, comprimento, altura/largura e um grau de confiança entre 0 e 1.

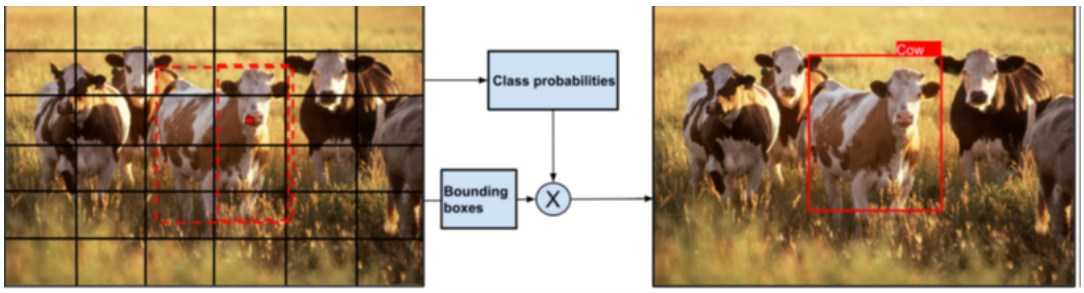


Figura 2.13: YOLO: Imagem em grelha SxS

2.3.1.2 YOLOv2

Com o intuito de melhorar o YOLO, Redmon et al. [33] propuseram um novo algoritmo a que chamaram YOLOv2. O YOLOv2 é uma versão melhorada do YOLO que pretende melhorar a velocidade de execução e a exatidão das localizações previstas.

A inovação mais impactante foi a adição de caixas âncora, como visto na arquitetura anterior o YOLO só consegue prever um objeto na mesma célula da grelha, o que introduz problemas quando existem mais do que um objeto na mesma célula. De forma a superar esta limitação o YOLOv2 introduz o conceito de âncora, que pode ser interpretado como uma sugestão de caixas circundantes. As caixas âncoras podem ter escalas e proporções diferentes entre si. No YOLOv2 em cada célula são usadas 5 caixas âncoras cujas dimensões são determinadas através do *cluster* k-means na fase de treino, deste modo as âncoras produzidas são mais adequadas aos dados.

O YOLOv2 tem como base a arquitetura de aprendizagem profunda Darknet-19 com a adição de 11 camadas dedicadas à deteção de objetos. Com um total de 30 camadas, YOLOv2 aumentou a velocidade e exatidão face à arquitetura anterior.

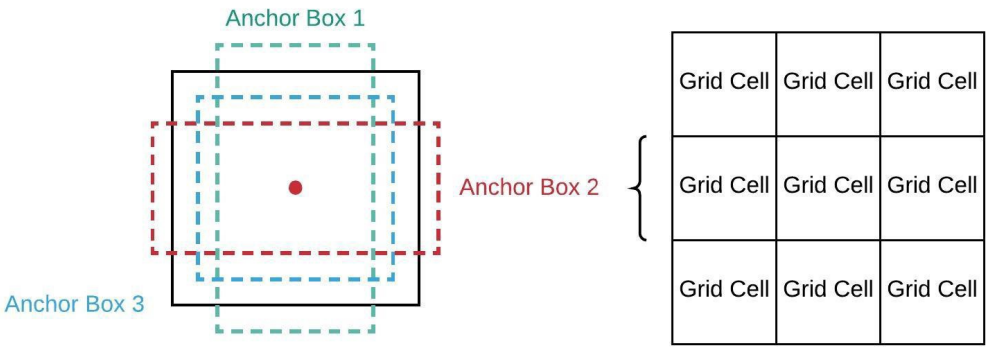


Figura 2.14: Caixas âncora

2.3.1.3 YOLOv3

O YOLOv3 [34] apresenta melhorias na exatidão apresentada pelo YOLOv2, mas com decréscimo na velocidade de execução. O YOLOv3 usa uma variante do Darknet que originalmente tem uma rede de 53 camadas às quais junta outras 53 dedicadas à deteção, o que explica a diminuição da velocidade de execução.

A inovação mais notável foi as caixas circundantes serem previstas em 3 escalas totalizando um total de 9 caixas âncora, o que permitiu que a versão 3 do YOLO aumentasse a capacidade de detetar objetos menores, algo que as versões anteriores do YOLO têm dificuldade.

2.3.1.4 YOLOv4

O YOLOv4 foi o proposto por Bochkovskiy et. al. [3] em 2020, o primeiro YOLO sem a participação dos autores originais. Os resultados apresentados foram superiores aos demais com uma AP de 45,3% sobre o MS COCO *dataset*, 10% superior ao YOLOv3. As principais inovações do YOLOv4 foram a adição de procedimentos de otimização aplicados durante a fase de treino e inferência.

2.3.2 *Tracking* de objetos

Os algoritmos de *tracking* têm como responsabilidade atribuir identidades únicas e manter a identidade constante aos objetos escolhidos para monitorizar na sequência de vídeo. Esta dissertação focou-se no algoritmo de *tracking* Deep SORT que é uma extensão do algoritmo SORT.

2.3.2.1 SORT

Simple Online and Real-Time Tracking, SORT é o algoritmo de *tracking* introduzido por Bewley et al. [2], SORT foi desenhado para monitorizar múltiplos objetos no paradigma de *tracking-by-detection*. O SORT é um algoritmo intencionalmente simples, pois é acompanhado por uma rede neuronal de deteção de objetos, assim sendo, o sucesso do *tracking* é dependente da exatidão do algoritmo de deteção.

A cada frame, o algoritmo SORT prevê a posição dos objetos aos quais já atribui uma identidade, a estimativa da posição do objeto é calculada através do Kalman filter com um modelo de velocidade linear. A seguir, o detetor de objetos deteta os objetos na frame corrente e o SORT compara a posição dada pelo detetor com a posição que previu, produzindo uma matriz de custos. A matriz de custo é calculada como IoU entre as deteções e as previsões. Por fim, as deteções da frame corrente são atribuídas aos objetos anteriormente identificados usando o Hungarian method. É atribuída uma identidade a um objeto quando este é detetado em várias frames consecutivas e não se sobrepõe a nenhum objeto já identificado.

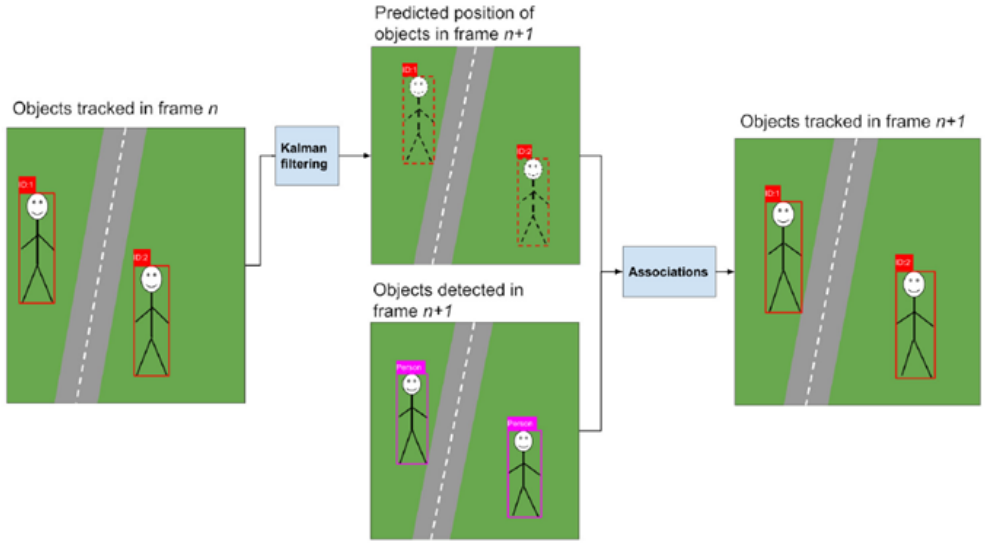


Figura 2.15: Esquema do algoritmo SORT

2.3.2.2 Deep SORT

O algoritmo Deep SORT [43] é uma extensão do algoritmo SORT, que pretende solucionar o problema das frequentes trocas de identidades presentes no SORT. O Deep SORT inova ao incorporar informação visual dos objetos detetados, aplicada no processo de coincidir os objetos anteriormente identificados com os presentes na frame corrente. A informação visual ou descritores de aparência são computados a partir de uma rede CNN pré-treinada com conjunto de dados relativos a reidentificação de pessoas. Ou seja, a rede neuronal está treinada a produzir descritores de aparência capazes de distinguir pessoas com base na aparência visual.

O Deep SORT utiliza 2 métricas de distância aquando da comparação entre as deteções efetuadas e os objetos já seguidos: distância Mahalanobis e a distância cosseno entre descritores de aparência. A distância Mahalanobis mede como a posição das deteções difere das posições dos objetos já seguidos em termos de desvios padrão e em relação à média dos objetos seguidos. Os descritores de aparência de cada nova deteção são comparados com os descritores dos objetos já seguidos, através do cálculo da distância cosseno entre eles. Com ambas as distâncias, é efetuada a atribuição das deteções correntes com os objetos já seguidos através do *Hungarian method*. Os descritores de aparência habilitaram o Deep SORT seguir os objetos em situações de oclusão por um número considerável de frames. A figura 2.16 apresenta resumidamente o funcionamento do algoritmo.

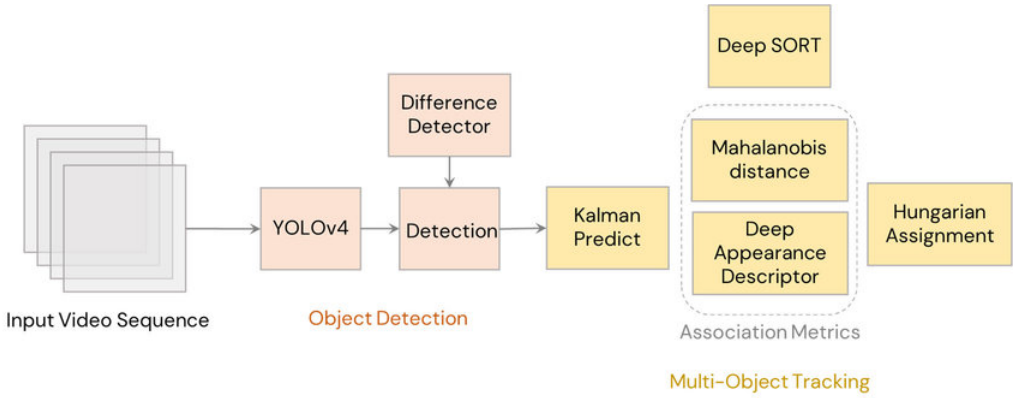


Figura 2.16: Esquema do algoritmo Deep SORT

2.3.3 Métricas de Avaliação

As métricas explicadas nesta secção foram enunciadas na secção 2.2.4 e contextualizadas para a temática de deteção e *tracking* de objetos.

2.3.3.1 VP, FP, FN em Deteção de objetos

O conceito de IoU é aplicado para determinar os VP, FP, FN gerados pelo detetor de objetos. Nas figuras abaixo estão presentes uma pessoa e um cavalo. Ignoraremos o cavalo e focamos-nos na pessoa. A azul está a caixa circundante verdadeira igual para todas as figuras e a laranja a caixa circundante prevista. Tipicamente quando a caixa prevista e a caixa verdadeira computam um valor de IoU superior a 0.5, a caixa circundante prevista é considerada como VP. As várias situações possíveis são:

Verdadeiro Positivo (IoU > 0,5)

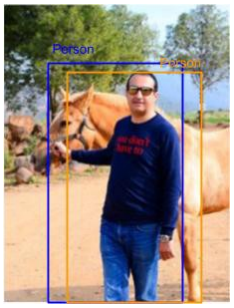


Figura 2.17: Exemplo de Verdadeiro Positivo [27]

Falso Positivo

A caixa circundante prevista é considerada FP se:

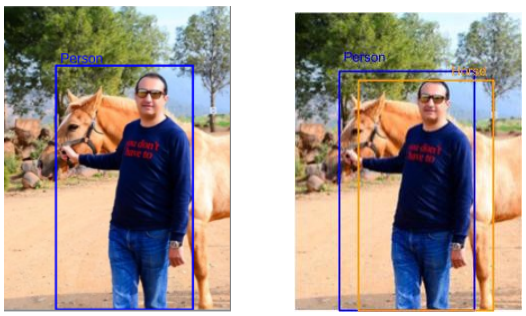
- IoU < 0.5
- Caixa circundante prevista duplicada

Figura 2.18: Exemplo de Falso Positivo [27]

Falso Negativo

A caixa circundante prevista é considerada FN se:

- Não for prevista nenhuma caixa circundante
- A previsão contém a classe errada



(a) (b)

Figura 2.19: Exemplo de Falso Negativo [27]

2.3.3.2 *Mean Average Precision*

A *Mean Average Precision* (mAP) é a métrica mais popular para medir a exatidão de detetores de objetos. A métrica AP ou AP@0.50 é calculada como a área abaixo da curva precisão abrangência [27]. O AP@0.50 significa que no cálculo da precisão e abrangência, o VP tem um valor mínimo IoU de 0.50. A mAP é a média de todas as AP começando na AP@0.50 até a AP@0.95, com o passo de tamanho 0.05.

2.3.3.3 Métricas *Multi-Object Tracking* (MOT)

As métricas associadas aos problemas de MOT são semelhantes aos problemas de Aprendizagem Automática, descritas na secção 2.2.4, com a adição da sigla ID antes do nome

da métrica [37].

$$\textbf{IDP} = \frac{IDVP}{IDVP + IDFP} \tag{2.6}$$

$$\textbf{IDA} = \frac{IDVP}{IDVP + IDFN} \tag{2.7}$$

$$\textbf{IDF1} = \frac{2IDVP}{2IDVP + IDFP + IDFN} \tag{2.8}$$

2.4 Artigos Relacionados

Os três artigos presentes nesta secção descrevem técnicas e paradigmas testados ou aplicados na dissertação. Os conceitos principais presentes nos artigos foram descritos nas secções anteriores.

2.4.1 Mapeamento vídeo de ténis para modelo 2D

No trabalho de Mora [29] é realizado o mapeamento entre a imagem de uma transmissão televisiva de uma partida de ténis para um modelo do campo 2D. O Padel tal como o ténis têm gravações de jogo semelhantes, nas quais é possível ver o campo por inteiro e numa posição vertical, o que por si entende uma projeção semelhante.

Para realizar o mapeamento é necessário encontrar os pontos a', b', c', d' marcados na figura 2.21 e o processo é descrito nos seguintes passos:

1. Extração dos píxeis brancos presentes na imagem que representam as linhas brancas limitadores do campo de ténis.

2. Aplicação da transformação Hough [30] de modo que os pixeis extraídos formem linhas potencialmente pertencentes ao campo.

3. As linhas detetadas são classificadas em horizontais e verticais. Uma linha é vertical se a distância entre as coordenadas y das extremidades for menor que a distância entre coordenadas x e horizontal, caso contrário.

4. As linhas verticais são ordenadas da esquerda para a direita e as horizontais de cima para baixo.

5. A cada par linha horizontal, linha vertical é calculada a interseção até que sejam computados os pontos a', b', c', d'.

Com os pontos a', b', c', d' e com os pontos a, b, c, d é calculada a matriz de transformação da perspetiva através da função *getPerspectiveTransform()* da biblioteca OpenCV [4]. Com a matriz de transformação da perspetiva é possível corresponder uma posição da imagem do jogo para o modelo 2D do campo e vice-versa, tal como ilustra a figura 2.21.

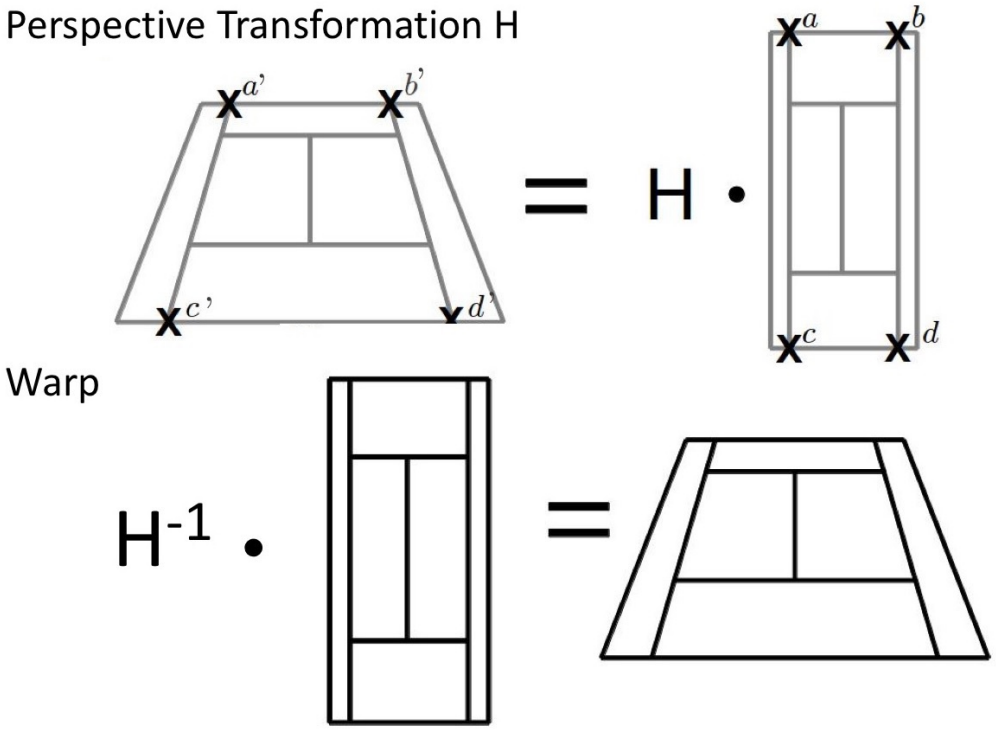


Figura 2.20: Aplicação da matriz de transformação de perspetiva

2.4.2 *Tracking* jogadores de andebol

O desporto de ténis tem como característica ter apenas um jogador em cada metade do campo, o que permite os jogadores serem constantemente visíveis. Os algoritmos de *tracking* normalmente aplicados ao ténis usam esta característica e assumem que o jogador numa metade do campo tem sempre a mesma identidade. No entanto, no desporto Padel, existem dois jogadores em cada metade do campo, o que torna necessário uma abordagem capaz de atribuir uma identidade única e constante a cada jogador. O algoritmo deve conseguir manter a identidade de um jogador em situações de oclusão parcial.

Nos artigos [19] e [7] está apresentada uma abordagem de deteção e *tracking* de jogadores de andebol. O andebol é um desporto de equipa onde os jogadores se movem rapidamente pelo campo todo, mudando constantemente de posições e frequentemente estão parcialmente ou completamente ocultados por outros jogadores.

O *tracking* de jogadores de andebol num vídeo é um problema de *Multi-Object Tracking*, onde o objetivo é seguir os vários objetos, neste caso jogadores, presentes num vídeo mantendo um único e constante ID para cada jogador. Esta é uma tarefa complicada, pois quantos mais jogadores a seguir mais subproblemas é preciso resolver, problemas como a oclusão, mudanças de direção abruptas e até semelhança de aparência.

O paradigma mais usado em problemas de *Multi-Object Tracking* é o *tracking-by-detection*. Neste paradigma o algoritmo de *tracking* depende do detetor de objetos para detetar e localizar o objeto na frame e o algoritmo de *tracking* fica com a tarefa de associar

as deteções entre frames consecutivas ao mesmo objeto. Para tal, o algoritmo de tracking usa as informações presentes nas caixas circundantes detetadas pelo detetor de objetos. Informações que podem ser dimensões, localização do centro, posição relativa nas frames anteriores e até características visuais da imagem.

No estudo efetuado foram usados como algoritmos de deteção de objetos, o YOLOv3 e o Mask R-CNN, como algoritmos de *tracking* foram usados o SORT e o Deep SORT. O Mask-RCNN foi pré-treinado com o conjunto de dados COCO[1] e o YOLO foi dividido em dois modelos, o primeiro treinado com os dados COCO e o segundo treinado com parte de um conjunto de dados de andebol, em ambos foi usada a técnica de transferência de conhecimento. Entre os modelos YOLO, o que foi treinado com os dados de andebol teve melhores resultados. Na comparação entre Mask-RCNN e YOLO, treinados com o conjunto de dados COCO, o Mask-RCNN teve um F1-score de 85% e o YOLOv3 teve 79%, ambos suficientemente bons para a análise de desempenho de jogadores. O YOLOv3 foi muito mais rápido que o Mask-RCNN, o que torna possível usar tanto em análise à posteriori como em tempo real. As métricas discutidas estão presentes na tabela 2.1.

Quanto aos resultados alcançados nos algoritmos de *tracking*, o Deep SORT apresenta melhores resultados que o SORT, ao adicionar características de aparência visual, ainda assim para cada jogador presente na frame e designado para *tracking* houve uma média de seis IDs diferentes. Nas conclusões dos artigos é ainda enunciado que o algoritmo de *tracking* tem debilidades em movimentos bruscos e na saída temporária e posterior reentrada dos jogadores anteriormente identificados.

Detetor de Objetos	Tempo/Frame	Precisão	Abragência	F1-Score
YOLOv3	0.04s	95%	68%	79%
Mask R-CNN	0.3s	98%	76%	85%

Tabela 2.1: Desempenho do YOLOv3 e Mask R-CNN

Figura 2.21: Deteção e *Tracking* de jogadores de andebol

[1] COCO é um conjunto de dados de grande escala para reconhecimento e segmentação de objetos [25].

2.4.3 Prever o sucesso de jogadas no ténis de mesa

O trabalho de Lukas Draschkowitz et al. [10] propõem um sistema capaz de prever o sucesso das jogadas/batidas na bola num jogo de ténis de mesa. A medida de sucesso adotada foi se a jogada a avaliar deu em ponto no instante a seguir.

O sistema de captura de imagem é composto por 2 câmaras colocadas em posições estratégicas, a primeira câmara é colocada no teto de forma a gravar o campo de cima, a segunda câmara é colocada na lateral da mesa para capturar a altura da bola em relação à mesa. A bola utilizada no jogo tem uma cor laranja forte que a distingue claramente do fundo. As bolas oficiais de ténis de mesa são de cor branca ou laranja, a bola laranja é mais facilmente detetada por ser menos influenciável ao efeito da luz.

A abordagem usada pelos autores do trabalho é dividida em duas grandes fases. A primeira é a fase de visão por computador, onde é realizada a deteção da bola e extração de *features*. A deteção da bola foi realizada através da deteção de pixeis de cor laranja. Com a informação da localização da bola sobre o tempo foram extraídas 8 *features* para cada jogada: que jogador bateu na bola, direção, velocidade e deslocamento da bola, toques, serviço, ponto e qualidade. A *feature* toques representa o número de toques de ambos jogadores até àquele momento. A qualidade é uma *feature* computada através da distância e velocidade da última pancada/jogada referente à descrita, somando os valores normalizados de ambos.

A segunda fase é a aplicação de Aprendizagem Automática. Após a recolha e construção do conjunto de dados ocorreu a divisão de dados de treino e teste com as proporções 66.6 % e 33.3 %, respetivamente. Os algoritmos aplicados foram *Sequential minimal optimization* [32], *Naive Bayes* [20], Árvore Aleatória [21] e Tabela de Decisão [23] da biblioteca Weka [11]. Os modelos apresentaram uma exatidão superior a 85% sobre o conjunto de teste. O algoritmo Árvore Aleatória considera n atributos aleatórios para cada nó. O processo de decisão pode ser facilmente seguido com uma representação visual, figura 2.22. A representação visual ajuda a extrair e interpretar possíveis padrões. No ramo direito da árvore é possível extrair alguns conselhos para os jogadores. No ramo esquerdo a extração de simples conselhos para os jogadores fica muito complicada devido à profundidade e número de nós da árvore.

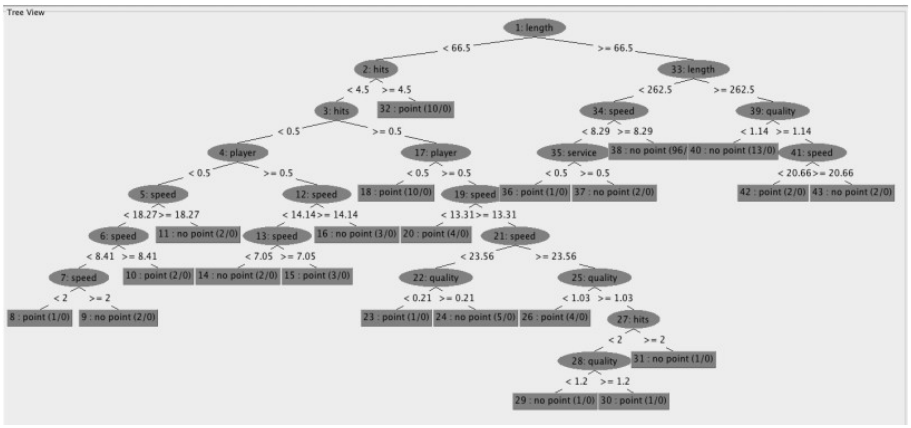


Figura 2.22: Representação Visual da Árvore Aleatória

3

Sistema de Visão

Este capítulo descreve a aplicação de técnicas de visão por computador em vídeos de sessões de Padel com foco na deteção e *tracking* dos jogadores.

3.1 Dados Iniciais

Uma das primeiras etapas do desenvolvimento desta dissertação foi perceber quais os dados digitais disponíveis e que informação se pretendia extrair no contexto desportivo do Padel.

O conjunto de dados utilizado foi disponibilizado pelo ABC Padel Indoor através do seu canal de YouTube[1] e contém transmissões de vídeo de partidas de Padel efetuadas nos campos do ABC. As sequências de vídeo variam em média entre uma hora a uma hora e meia e tem uma taxa de 30 frames por segundo. A resolução dos vídeos é 1280 x 720 píxeis. As gravações dos vídeos foram realizadas com uma câmera que captura o campo de Padel em toda a sua plenitude e numa posição vertical fixa. Além do campo, estão presentes duas equipas de dois jogadores em metades opostas e uma bola. Os jogadores não têm numeração que os identifique, prática comum em desportos de equipa, e por vezes têm cor de roupa semelhante. Na figura 3.1 está presente uma frame de um dos vídeos.

Antes de começar o Treinador Virtual foram levantados requisitos do que era necessário para o sucesso do mesmo. Os requisitos apurados foram ser necessário mapear a imagem do campo para um modelo 2D, detetar e atribuir uma identidade a cada jogador e por fim detetar os momentos em que o jogador toca na bola. A informação recolhida será utilizada na avaliação das jogadas do jogador na componente do Treinador Virtual.

[1] https://www.youtube.com/c/worldpadeltv

Figura 3.1: Frame de um vídeo de Padel do ABC

3.2 Mapeamento do campo

O campo de Padel tem dimensões de 10 metros de largura por 20 metros de comprimento. Na secção 2.4.1 foi descrito o mapeamento do campo de ténis e a procura dos 4 pontos necessários para o cálculo da matriz de transformação da perspetiva. Após alguns testes para detetar automaticamente os 4 pontos, tanto através da estrutura metálica que limita o campo como com a deteção dos limites da cor dominante no campo, foi decidido que os 4 pontos eram manualmente inseridos e a seguir calculada a matriz de transformação da perspetiva, através da função *getPerspectiveTransform()* da biblioteca OpenCV [4]. Esta decisão foi tomada porque diferentes campos de Padel têm uma cor dominante diferente e assim torna mais versátil o mapeamento do campo. A câmara de vídeo ao estar fixa permite que o mapeamento do campo seja efetuado apenas uma vez. O modelo do campo 2D está presenta na figura 3.6, as distâncias nos eixos representam aproximadamente as dimensões do campo em cm.

3.3 Deteção e *Tracking* dos jogadores

Nos vídeos das partidas estão presentes 4 jogadores de Padel passíveis de serem detetados e identificados. Este problema é um problema MOT e foi seguido o paradigma *tracking-by-detection* para o resolver. O paradigma *tracking-by-detection* é o mais utilizado para resolver problemas de MOT e consiste em aplicar inicialmente um detetor de objetos e a seguir um algoritmo de tracking [5].

O detetor de objetos escolhido foi o YOLOv4, descrito na secção 2.3.1.4. O YOLOv4 é um algoritmo ideal para este problema por ser relativamente rápido e fiável. O detetor de objetos foi treinado com transferência de conhecimento em parte de conjunto de dados de Padel. O conjunto de dados tem 320 imagens anotadas manualmente sobre os dados inicias apresentados. O conjunto de dados foi anotado com uma classe, *player*, segundo as regras do protocolo padrão descrito em [39]. As anotações foram realizadas com a ferramenta labelImg[2] presente na figura 3.2. O YOLO foi treinado com 70% do conjunto de dados por 1000 iterações e contabilizou uma mAP de 99.82% sobre os restantes 30%. A figura 3.4 mostra os resultados do detetor de objetos numa frame.

O algoritmo de *tracking* aplicado foi o Deep SORT, descrito na secção 2.3.2.2. O Deep SORT é um algoritmo interessante porque utiliza descritores visuais no processo de *tracking*. Os descritores visuais são computados através de uma rede CNN pré-treinada com conjunto de dados de reidentificação de pessoas. Os descritores de aparência são comparados através da distância cosseno. O Deep SORT tem parâmetros passíveis de manipulação que são o parâmetro de iniciação, número de frames detetadas consecutivamente antes de atribuir uma identidade, idade, por quantas frames é guardada uma identidade não presente nas frames atuais e a máxima distância IoU. A melhor combinação de parâmetros foi iniciação igual a 45, idade igual a 100 e IoU igual a 0.80. A aplicação do Deep SORT foi testada no vídeo que tem a duração de 9 min e 30 segundos e obteve um IDP de 94.4%, IDA de 99.4% e IDF1 de 96.8%. A combinação de resultados e parâmetros está presente na tabela 3.1. No vídeo testado estão presentes 4 jogadores constantemente no recinto desportivo. A aplicação do algoritmo resultou em 5 identidades atribuídas, ou seja, ocorreu uma mudança de identidade. Na figura 3.5 está a imagem obtida pelo detetor de objetos juntamente com a informação do algoritmo de *tracking* que atribui um ID a cada jogador.

A implementação do detetor de objetos YOLOv4 utilizada foi AlexeyAB YOLO[3] e o do algoritmo de *tracking* foi do Nicolai Wojke Deep SORT[4]. As implementações escolhidas são da autoria dos autores originais dos artigos referentes.

[2] https://github.com/tzutalin/labelImg
[3] https://github.com/AlexeyAB/darknet
[4] https://github.com/nwojke/deep_sort

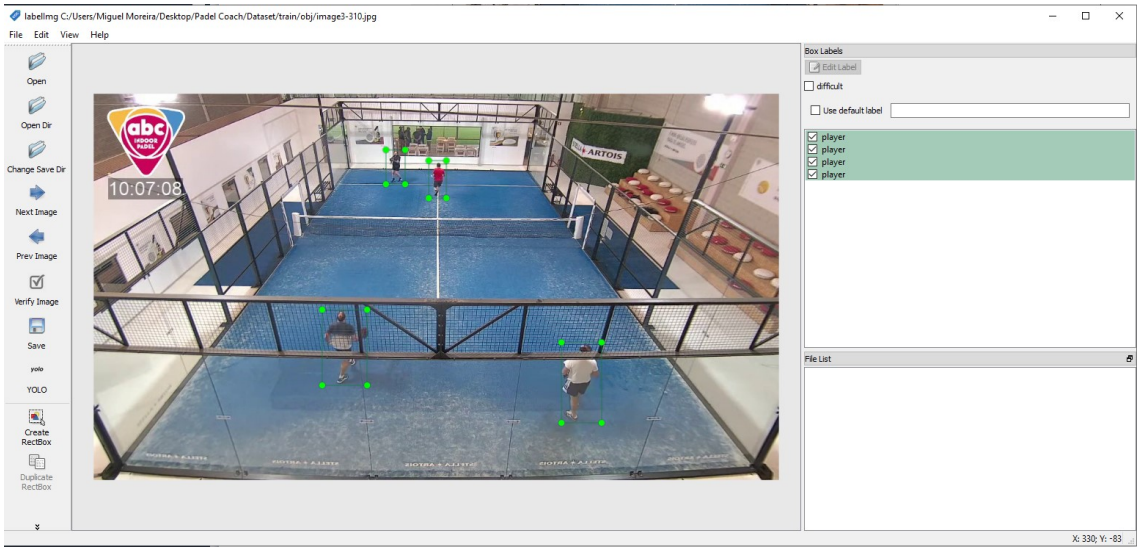


Figura 3.2: LabelImg : Anotação de jogadores

Iniciação	Idade	IoU	IDP	IDA	IDF1
30	100	0.7	93.8	98.8	96.3
45	100	0.7	93.8	99.1	96.4
30	100	0.8	93.8	98.8	96.3
45	100	0.8	93.8	99.1	96.4
30	150	0.8	94.0	99.5	96.6
45	150	0.8	94.4	99.4	96.8

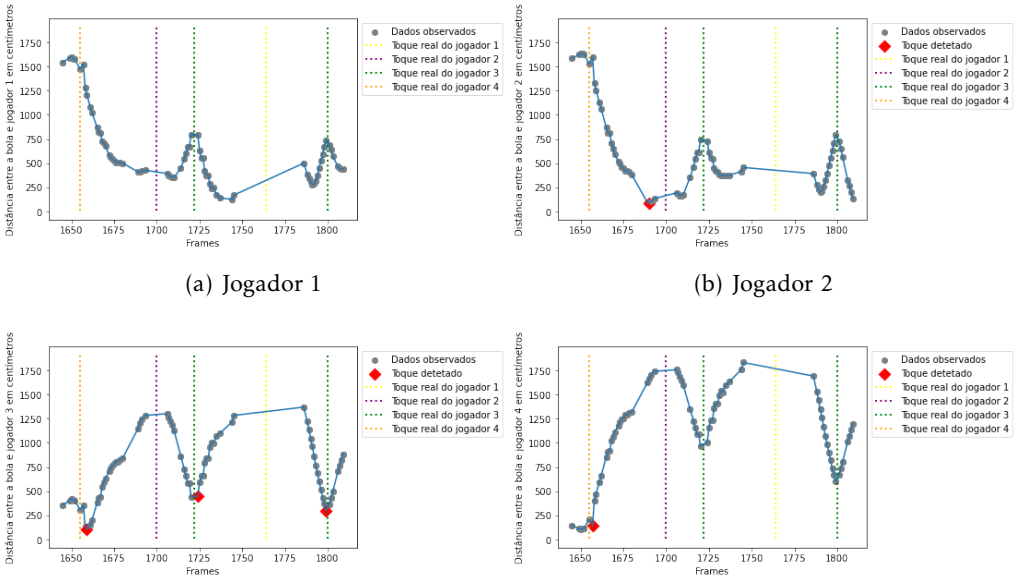
Tabela 3.1: Parâmetros e Métricas do algoritmo Deep SORT

3.4 Momentos de jogada

Após a deteção e *tracking* dos jogadores presentes, o passo a seguir é detetar em que momentos durante o vídeo aconteceram jogadas. Como jogadas é considerado os momentos em que o jogador toca na bola com o intuito de a mover até ao campo adversário.

Este ensaio recorre à posição dos diferentes jogadores, obtida na secção anterior e à posição da bola proveniente de um projeto paralelo sobre a deteção da bola com base no artigo Tracknet[17]. O gráfico 3.3 mostra a distância euclidiana entre a bola e cada jogador sobre as frames que representam o intervalo temporal. Nos gráficos é possível constatar que os mínimos locais no eixo y são os momentos em que a bola está mais próxima dos jogadores e que tendencialmente podem ser interpretados como toques dos jogadores na bola. Nos gráficos está apresentado, com uma linha vertical tracejada, os momentos em que os jogadores verdadeiramente tocam na bola e com um losango vermelho são representados os momentos em que são detetados toques. No gráfico relativo ao jogador 1 não é detetado nenhum toque pela falta de dados observados no momento do toque real, resultando num falso negativo. No gráfico do jogador 3 são detetados 3 toques, no entanto, o jogador 3 apenas toca 2 vezes na bola, ou seja, o primeiro toque detetado é

um falso positivo. Os restantes toques dos jogadores foram detetados e sintetizados na tabela 3.2. O jogador 3 e 4 têm mínimos locais por volta da frame 1726 e 1800. Como é impossível os dois jogadores da mesma equipa terem tocado a bola no mesmo instante, o toque é atribuído ao jogador mais próximo da bola.



(a) Jogador 1

(b) Jogador 2

(c) Jogador 3

(d) Jogador 4

Figura 3.3: Distância entre a bola e Jogadores

Jogador	2	3		4
Toque real	1700	1722	1800	1655
Toque detetado	1686	1724	1799	1658
Erro(frames)	14	2	1	3

Tabela 3.2: Erro entre toque detetado e real

Com o intuito de explorar e validar a proposta feita foi realizado o mesmo teste num vídeo com 6887 frames e com um total de 48 toques. Um toque detetado foi considerado Verdadeiro Positivo se a diferença entre o toque verdadeiro e o detetado é menor que a margem de 30 frames. O vídeo contabilizou 37 VP, 32 FP e 8 FN. Algumas das razões que justificam o elevado número de falsos positivos são ressaltos da bola no vidro e a trajetória da bola passar próxima de ambos jogadores da mesma equipa. Assim sendo foi decidido não prosseguir com esta ideia e é proposto como trabalho futuro uma abordagem de reconhecimento de ação para detetar os momentos em que aconteceram toques na bola.

3.5 Conclusões

Com a informação obtida na secção 3.3 e com a matriz de transformação de perspetiva calculada na secção 3.2 é possível representar a posição dos jogadores num modelo do campo 2D, figura 3.7. A posição na caixa circundante escolhida para representar o jogador num modelo 2D é a posição centro inferior que é a posição mais próxima dos pés dos jogadores. Porém, existe uma discrepância entre as posições de referência dos jogadores da metade superior e inferior do campo. A discrepância é que os jogadores na metade inferior têm a posição de referência mais perto da linha final do campo e os jogadores na metade superior têm a posição de referência mais longe da linha final do campo. Esta discrepância é consequência da utilização de apenas uma câmara na gravação dos vídeos.

A realização e estudo dos requisitos permitiu perceber a informação que é passível de ser extraída com sucesso dos vídeos e irá justificar questões de desenho na secção seguinte. Por fim, os dados obtidos na deteção e *tracking* dos jogadores são guardados em formato JSON e são fonte de dados para a futura aplicação com que os utilizadores vão interagir.

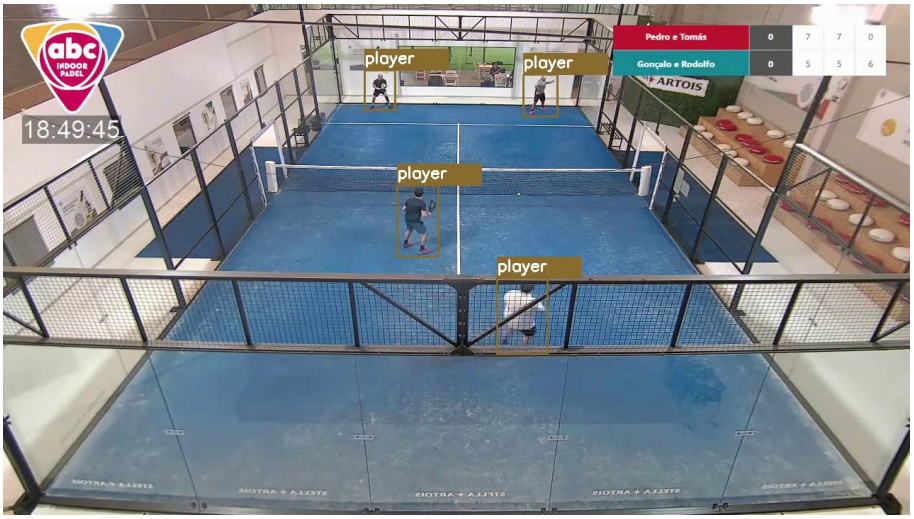


Figura 3.4: Resultado do detetor de objetos

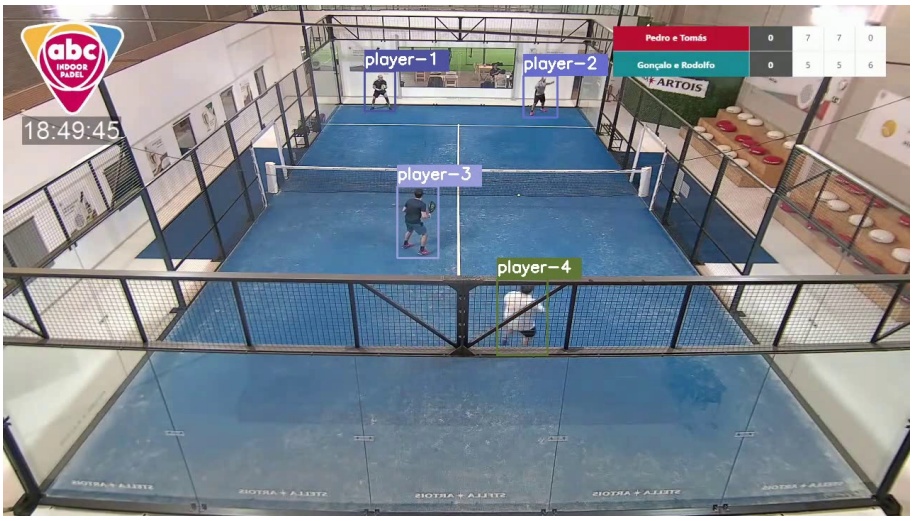


Figura 3.5: Deteção dos jogadores mais ID atribuído

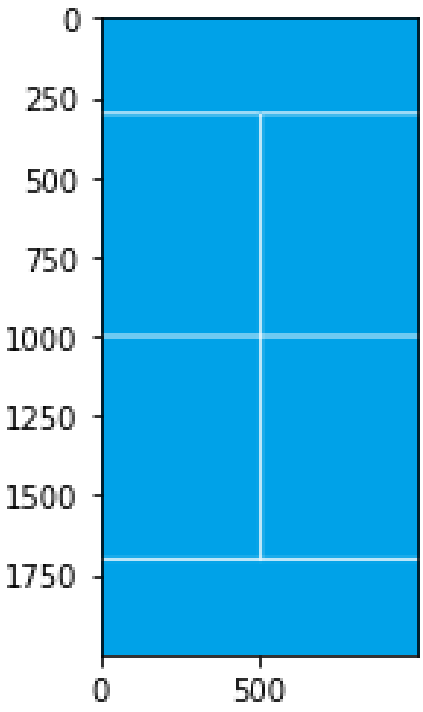


Figura 3.6: Modelo do Campo 2D

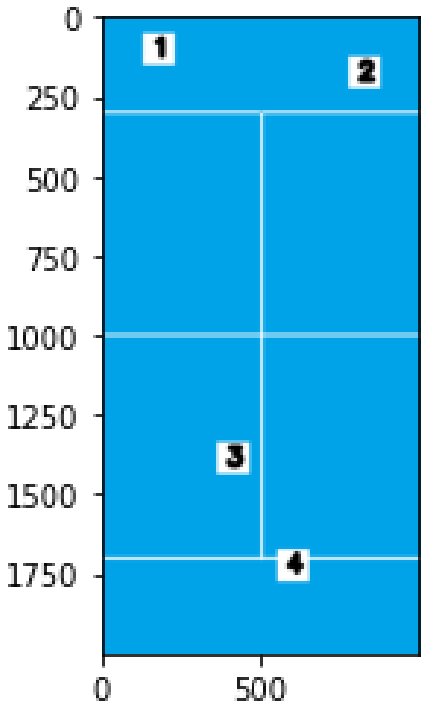


Figura 3.7: Posição dos jogadores no Campo 2D

4

Metodologias para a solução

Após o desenvolvimento do Sistema de Visão que permitiu extrair informação sobre a posição relativa dos jogadores, segue-se este capítulo que descreve a metodologia efetuada que culminou no desenvolvimento do Treinador Virtual.

4.1 Recolha de dados

O primeiro passo no desenvolvimento do Treinador Virtual passou por recolher um conjunto de dados, visto não haver no domínio público conjuntos de dados sobre jogadas de Padel. Um pequeno protocolo experimental foi escrito para regularizar o conjunto de dados necessário para este estudo. Esta secção explica os passos presentes no protocolo e seguidos no processo de recolha de dados.

Antes de começar a recolha de dados, foi desenvolvida uma ferramenta que auxiliou no processo de anotação. A ferramenta funciona da seguinte maneira, dado um vídeo extrai todas as frames e numa sequência de cliques permite anotar a posição da bola, jogadores e o local ideal para onde a bola deve ser jogada. Os dados anotados são guardados com um ID de cada frame num ficheiro CSV.

O passo seguinte foi a seleção de vídeos para anotação. Os vídeos escolhidos para anotação foram os do ABC, descritos na secção 3.1, foram selecionados 13 vídeos com a duração média de 10 minutos. Todos os vídeos contêm duplas de jogadores diferentes e todos os jogadores seguram a raquete com a mão direita. A mão predominante de cada jogador influencia a jogada ideal em cada situação, assim sendo foi escolhido somente anotar jogadores destros por serem a maioria a nível global.

O processo de anotação foi realizado em três fases:

1. A primeira fase consistiu em escolher as imagens em que um jogador dá uma pancada na bola. Nas imagens escolhidas é anotada a posição da bola (círculo vermelho), a posição dos 4 jogadores com a seguinte ordem: jogador que lançou a bola (**Jogador 1**), colega de equipa (**Jogador 2**), jogador adversário mais à esquerda (**Jogador 3**) e jogador adversário mais à direita (**Jogador 4**) e por fim se a pancada é um serviço.

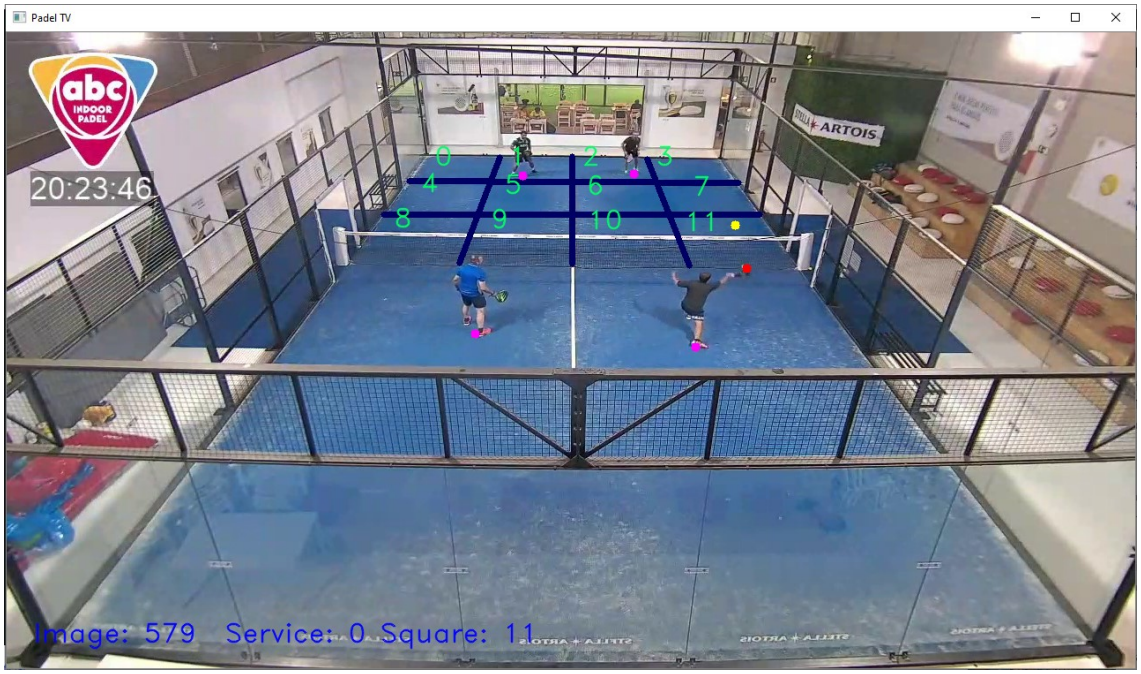


Figura 4.1: Ferramenta de anotação

O local na imagem onde é anotada a posição do jogador tenta replicar o comportamento do detetor de objetos, ou seja, na metade superior é anotado o centro entre os dois pés e na metade inferior é anotado o centro da última parte do corpo. Este conjunto de atributos foi escolhido por conseguirem ser extraídos pelo Sistema de Visão montado com exceção ao atributo serviço. Este foi adicionado porque se a jogada é um serviço, a bola só pode ser jogada para uma determinada zona do campo.

2. Na segunda fase, as anotações realizadas foram dirigidas a dois especialistas que adicionaram a informação do melhor sítio para onde jogar (círculo amarelo). Os especialistas são dois jogadores de Padel experientes ao nível de competição. O processo de anotação começou com os especialistas a anotar um pequeno conjunto de 20 amostras. Os especialistas apresentaram concordância neste conjunto que justificou que o restante conjunto de dados fosse anotado de modo mutuamente exclusivo pelos especialistas. Esta escolha permitiu a anotação mais rápida do conjunto de dados.

3. Por fim, às anotações numéricas das posições de cada jogador é aplicada a matriz de transformação de perspetiva descrita na secção 3.2.

As anotações realizadas têm como princípio que o jogador que bate a bola está sempre na metade inferior do campo. Quando o jogador que bate a bola está na metade superior é realizada a inversão do campo, ou seja, os jogadores da metade superior são movidos para metade inferior e os da metade inferior para a metade superior.

4.2 Processamento das anotações

Após a recolha dos dados e com o *feedback* dos especialistas foram realizadas duas alterações aos dados anotados. A primeira alteração consistiu em remover o atributo correspondente à posição da bola. Esta alteração foi motivada pela frequente oclusão da bola perante a raquete. Esta situação dificultaria a futura extração deste atributo.

A segunda alteração é relativa à abordagem ao problema. Os especialistas ao anotarem o melhor local para onde jogar tiveram dificuldades em escolher apenas um local exato. O *feedback* recebido foi que o local anotado era uma jogada ideal, como também seria um conjunto de locais próximos ao anotado. Assim sendo foi decidido aglomerar esses locais e tratar a procura da melhor jogada como um problema de classificação, onde cada classe representa uma região do campo. Em conjunto com os especialistas procurou-se definir as tais regiões. As regiões foram definidas com auxílio das linhas do campo existentes.

Cada metade do campo de Padel tem por si mesmo 2 linhas: a linha que divide verticalmente o campo e a linha horizontal dos 3 metros. Na horizontal foi inserida uma linha imaginária a dividir igualmente 7 metros. Na vertical foram inseridas duas linhas imaginárias entre os limites verticais do campo e a linha do meio. As interseções entre linhas e limites do campo formam as regiões que se podem observar na figura 4.2.

Considerando como ponto referencial o canto superior esquerdo, as regiões são definidas como:

- **Região 0**: Varia em largura entre 0 e 2,5 metros e comprimento entre 0 e 3 metros.
- **Região 1**: Varia em largura entre 2,5 e 5 metros e comprimento entre 0 e 3 metros.
- **Região 2**: Varia em largura entre 5 e 7,5 metros e comprimento entre 0 e 3 metros.
- **Região 3**: Varia em largura entre 7,5 e 10 metros e comprimento entre 0 e 3 metros.
- **Região 4**: Varia em largura entre 0 e 2,5 metros e comprimento entre 3 e 6,5 metros.
- **Região 5**: Varia em largura entre 2,5 e 5 metros e comprimento entre 3 e 6,5 metros.
- **Região 6**: Varia em largura entre 5 e 7,5 metros e comprimento entre 3 e 6,5 metros.
- **Região 7**: Varia em largura entre 7,5 e 10 metros e comprimento entre 3 e 6,5 metros.
- **Região 8**: Varia em largura entre 0 e 2,5 metros e comprimento entre 6,5 e 10 metros.
- **Região 9**: Varia em largura entre 2,5 e 5 metros e comprimento entre 6,5 e 10 metros.
- **Região 10**: Varia em largura entre 5 e 7,5 metros e comprimento entre 6,5 e 10 metros.
- **Região 11**: Varia em largura entre 7,5 e 10 metros e comprimento entre 6,5 e 10 metros.

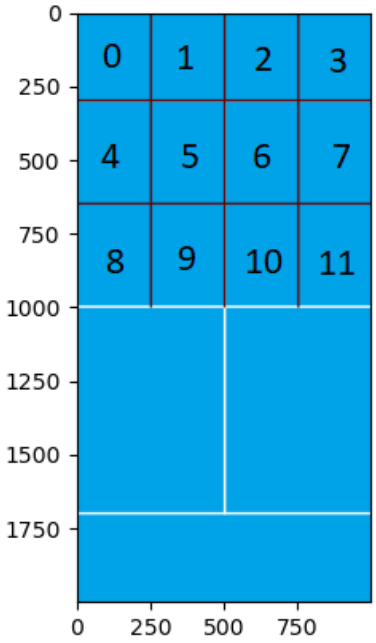


Figura 4.2: Limites das Regiões definidas

Com o intuito de validar as 12 regiões definidas foi computado o KMeans com 12 clusters da informação da melhor localização para onde jogar a bola. A figura 4.3 mostra os conjuntos de pontos propostos pelo KMeans que coincidem aproximadamente com os limites gráficos das 12 regiões propostas. Os números na legenda da figura não representam as regiões anteriormente definidas.

4.3 Análise do conjunto de dados

O sucesso da aplicação de Aprendizagem Automática num problema depende tanto dos dados como a escolha do algoritmo. O conjunto de dados anotado contém 1011 amostras e cada amostra define a jogada ideal no momento em que o jogador joga a bola. Cada amostra tem 8 *features* numéricas, 1 *feature* lógica e uma classe/região. As *features* numéricas descrevem as coordenadas x e y dos jogadores, a *feature* lógica indica se a jogada foi um serviço. Na tabela 4.4 estão presentes uns exemplos das amostras.

A figura 4.5 mostra a distribuição espacial das *features* numéricas para cada jogador, como esperado as posições do jogador 1 e jogador 2 estão distribuídas por toda a metade inferior do campo. O jogador 3 tem as anotações do lado superior esquerdo e o jogador 4 do lado superior direito. A análise da distribuição é importante para validar o conjunto de dados anotado. Quanto à *feature* do serviço a distribuição quantitativa está presente na figura 4.6.

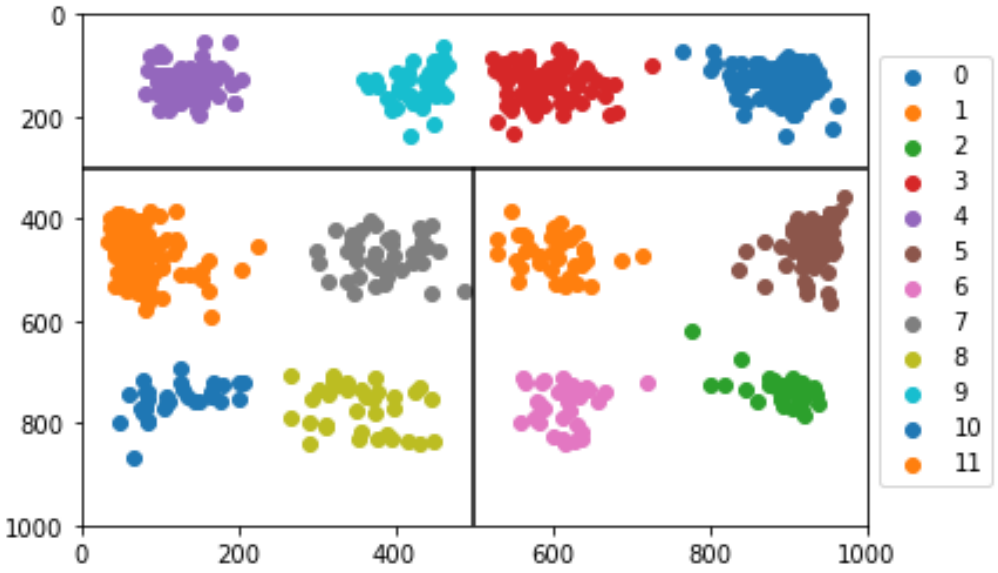
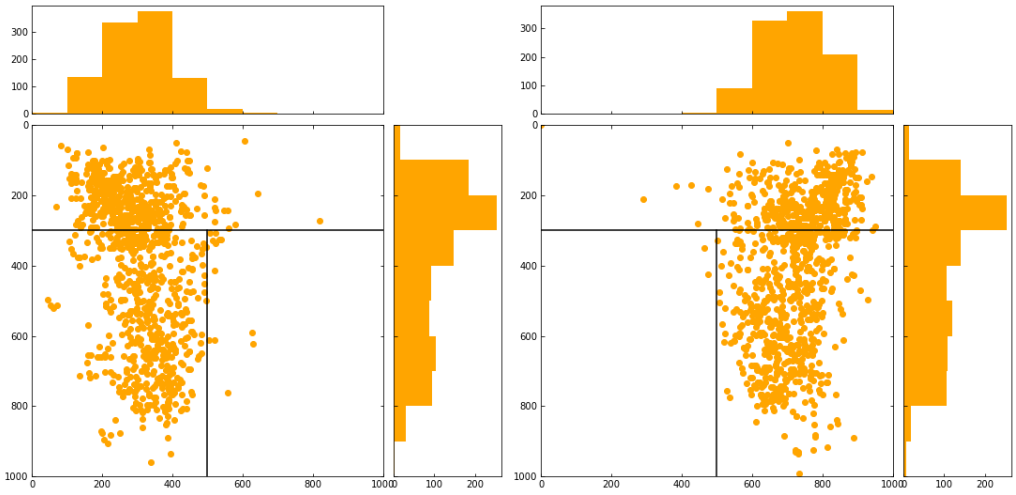


Figura 4.3: KMeans proposto para 12 clusters

Jogador1_X	Jogador1_Y	Jogador2_X	Jogador2_Y	Jogador3_X	Jogador3_Y	Jogador4_X	Jogador4_Y	Serviço	Região
719	1211	207	1771	356	304	773	502	0	1
375	1890	776	1295	342	345	747	578	0	3
341	1833	771	1328	300	233	886	218	1	7

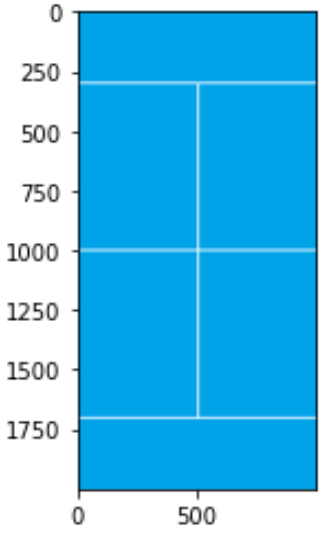
Figura 4.4: Exemplo dos dados anotados

A distribuição das classes das amostras está ilustrada no gráfico de barras 4.7. No gráfico é possível verificar que certas classes são mais numerosas que outras. Classes como as 3, 4, 7 são 3 vezes mais numerosas que as classes 8, 9, 10, 11. O desequilíbrio na distribuição de classes é normal acontecer, pois, jogar a bola para certas regiões é mais frequente do que para outras. As mecânicas do desporto favorecem jogadas nas zonas próximas dos limites do campo, visto que a bola tende a ressaltar do chão e bater no vidro, tomando uma trajetória aleatória.

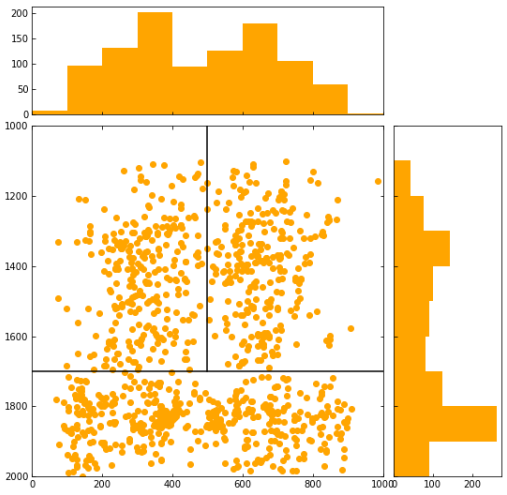


(a) Distribuição do jogador 3

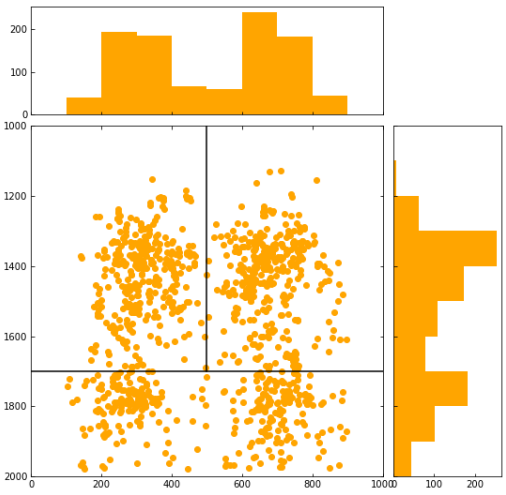
(b) Distribuição do jogador 4



(c) Modelo do campo 2D



(d) Distribuição do jogador 1



(e) Distribuição do jogador 2

Figura 4.5: Distribuição das anotações relativas às posições dos jogadores

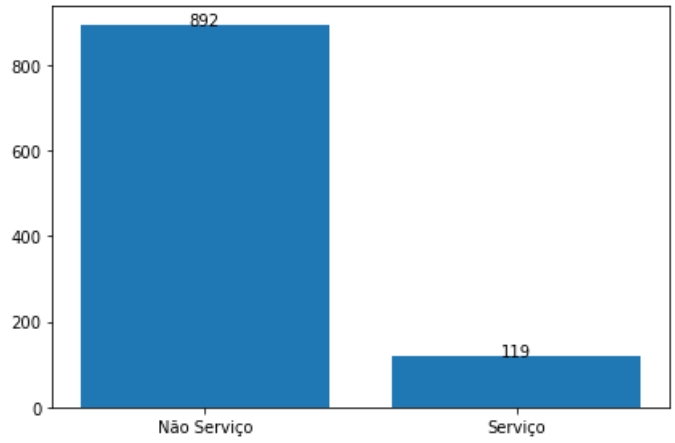


Figura 4.6: Gráfico de barras relativo à distribuição da característica Serviço

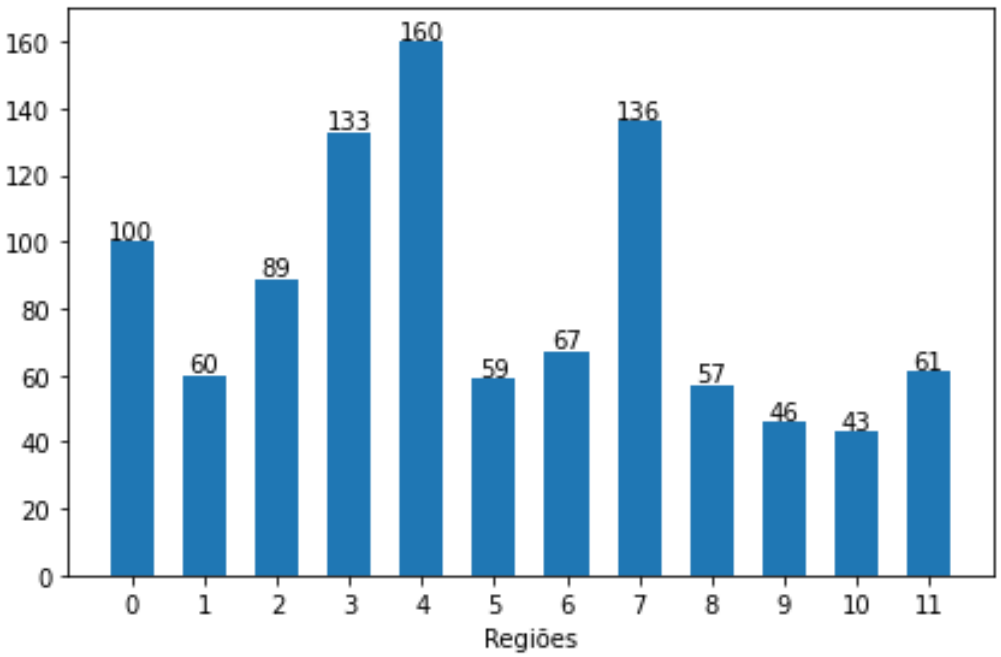


Figura 4.7: Gráfico de barras relativo à distribuição da Classe das anotações

4.4 Normalização e divisão de dados

A normalização é uma técnica frequentemente aplicada na preparação dos dados para os algoritmos de Aprendizagem Automática. A normalização visa mudar o valor dos dados numéricos para uma escala comum sem distorcer as diferenças nos intervalos de valores. A aplicação da normalização revela-se especialmente importante nos algoritmos que calculam a distância entre pontos. As distâncias são calculadas com base nos atributos e caso estes não tenham a mesma escala, os cálculos tendem a ficar mais complexos e demorados.

A normalização Min-Max é uma das normalizações mais comuns e transforma os atributos para pertencerem à escala de valores entre 0 a 1. Os valores são transformados pela seguinte fórmula:

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{4.1}$$

Onde x é o valor original e o x' é o valor normalizado.

Os algoritmos K vizinhos mais próximos e Support Vector Machine receberam dados normalizados com o método Min-Max enquanto os algoritmos Florestas Aleatórias e Árvore de Decisão receberam os dados sem a normalização.

A divisão do conjunto de dados em subconjuntos de treino e teste é um dos procedimentos mais comuns para avaliar o acerto do modelo de aprendizagem. O subconjunto de treino contém os dados rotulados para que o modelo consiga aprender sobre os dados e correspondentes classes e depois generalizar para dados futuros. O subconjunto de teste contém os dados sem as classes e procura testar e extrair as métricas do modelo, descritas e enumeradas na secção 2.2.4.

Uma abordagem comum é reservar parte dos dados de treino num subconjunto chamado dados de validação. Os dados de validação são usados visando avaliar imparcialmente o ajuste de parâmetros do modelo aos dados de treino e ajustar os hiperparâmetros. O ciclo de treino nesta abordagem é treinar o modelo com os dados de treino, ajustar os hiperparâmetros através dos dados de validação e por fim fornecer uma avaliação final e imparcial do modelo com os dados de teste. Esta abordagem tem como desvantagem reduzir o número de dados de treino, a partir dos quais os modelos aprendem a generalizar.

A abordagem utilizada foi a Validação Cruzada (VC) e são usados os subconjuntos de dados de treino e teste. A Validação Cruzada divide os dados de treino num número arbitrário de subconjuntos. O modelo é então treinado com os subconjuntos gerados menos um. O subconjunto restante é usado para validar e calcular uma medida de desempenho. É realizado um ciclo em que todos os subconjuntos em iterações diferentes são usados para validação. A medida de desempenho é calculada como a média dos desempenhos de cada iteração. Esta abordagem é especialmente importante no contexto do nosso problema porque a distribuição de classes do conjunto de dados utilizado é desequilibrada.

O conjunto de dados original foi primeiramente baralhado e depois dividido. A divisão dos dados em subconjuntos de treino e teste foi realizada com as proporções de 70% para treino e 30% para teste. A maior proporção de dados de treino permite que o modelo possua mais dados para aprender e que idealmente generalize melhor futuros dados.

4.5 Sobre amostragem do conjunto de dados

A distribuição quantitativa dos dados influencia o desempenho dos modelos. Como visto na secção 4.3, a distribuição dos dados presentes é desequilibrada, algumas classes são três vezes mais numerosas que outras.

Em problemas onde o conjunto de dados é desequilibrado, os algoritmos de AA tendem a negligenciar as classes menos numerosas. Isto acontece por os algoritmos serem desenhados a maximizar a exatidão e reduzir os erros, e se há muitas mais amostras de uma classe, o algoritmo foca-se a acertar nas mesmas, que por sua vez reduz os erros e melhora a exatidão de modo geral.

As técnicas de reamostragem pretendem a minimizar este problema. As duas técnicas mais populares são:

- **Sub-amostragem:** Técnica em que são removidas as amostras das classes maioritárias com o objetivo de equilibrar o conjunto de dados. A quantidade de dados que se pretender remover pode ser definida a indicar as proporções entre classes maioritárias e minoritárias. A sub-amostragem tem como desvantagem a perda de informação resultante da remoção aleatória de amostras.

- **Sobre-amostragem:** Técnica em que é aumentado o número de amostras das classes minoritárias com o objetivo de equilibrar o conjunto de dados. Uma das técnicas mais simples de sobre-amostragem é a duplicação dos dados em minoria.

O conjunto de dados anotado não é extenso com classes com apenas quarenta e poucos elementos. Assim sendo foi decidido aplicar a técnica de sobre-amostragem. A sobre-amostragem foi aplicada após a divisão do conjunto de dados e somente no subconjunto de treino. A distribuição dos dados de treino antes e depois da sobre-amostragem estão presentes nas figuras 4.8 e 4.9 respetivamente. A técnica de sobre-amostragem aplicada foi a SMOTE [8].

4.6 Modelos e Parâmetros

A escolha de algoritmos e parametrização dos mesmos é um passo importante na resolução de um problema de AA. Diferentes algoritmos pressupõem comportamentos diferentes no processo de classificação. Os parâmetros ajustáveis dos mesmos também afetam o resultado da classificação. Os dois subcapítulos abaixo indicam os algoritmos escolhidos e como foi realizada a parametrização.

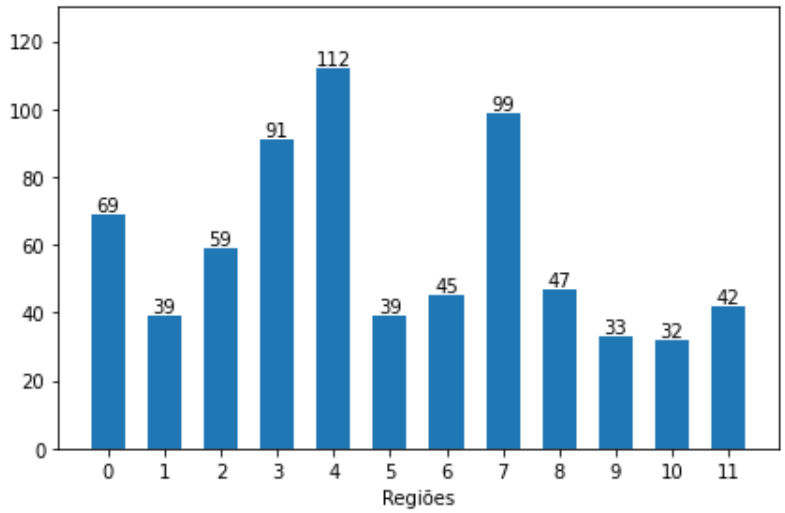


Figura 4.8: Distribuição original das regiões/classes do conjunto de treino

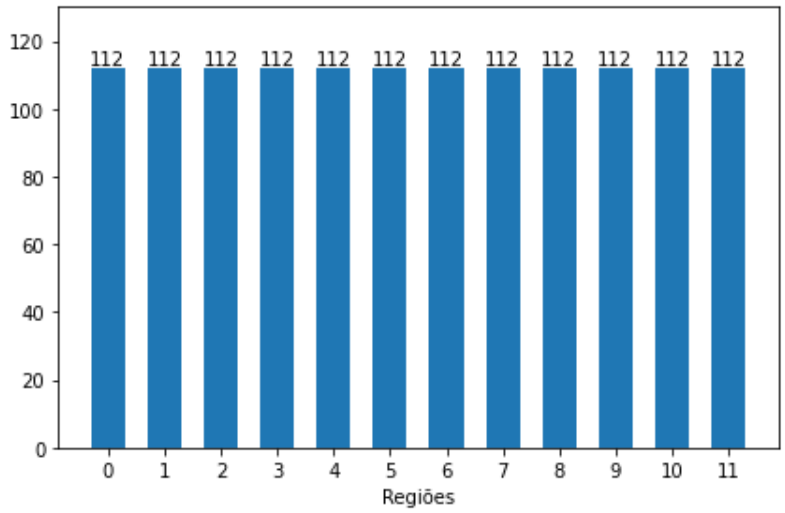


Figura 4.9: Distribuição das regiões/classes do conjunto de treino após sobre-amostragem

4.6.1 Algoritmos

Os algoritmos escolhidos e parametrizados foram Árvore de Decisão, Florestas Aleatórias, *Support Vector Classifier* que é o classificador Support Vector Machine com parâmetro de regularização C e o algoritmo K vizinhos mais próximos. A escolha destes algoritmos tem como base pertencerem a conjuntos de aprendizagem diferentes e serem apropriados para conjuntos de dados pequenos.

A Árvore de Decisão é um algoritmo simples onde é possível extrair regras e consegue ser visualmente fácil de interpretar. A FA é um algoritmo baseado na aprendizagem por conjuntos e desperta interesse o desempenho do mesmo na resolução deste problema. O SVM é um algoritmo bastante utilizado em trabalhos de classificação. Por fim, o KNN é um algoritmo simples de interpretar e robusto sobre dados ruído presentes na fase de

treino.

4.6.2 Parametrização

Os modelos de AA precisam que sejam definidos e ajustados os seus hiperparâmetros de forma a maximizar o desempenho dos modelos. Os hiperparâmetros são definidos como características do modelo que não podem ser estimadas a partir dos dados, assim sendo têm de ser definidos antes da fase de treino. Um exemplo de hiperparâmetro no KNN é o número de vizinhos com os quais queremos contar. Os hiperparâmetros dos algoritmos considerados estão presentes na tabela 4.1 com os seus respetivos valores padrão.

A abordagem utilizada para encontrar os melhores hiperparâmetros de cada modelo foi a utilização do algoritmo *Grid Search*. O *Grid Search* começa por criar um dicionário com todos os hiperparâmetros e respetivos conjuntos de valores que se pretende testar. O algoritmo testa todas as combinações possíveis de parâmetros durante a fase de treino e retorna a combinação de parâmetros que obteve melhor desempenho. O desempenho é medido através do procedimento de VC referido na secção 4.4. A tabela 4.2 indica os parâmetros e o intervalo de valores usados na otimização dos hiperparâmetros.

Os modelos foram parametrizados com e sem a aplicação da técnica de sobre-amostragem com o intuito de comparar a efetividade da técnica na resolução do problema desta dissertação.

Algoritmo	Hiperparâmetro	Valor
Árvore de Decisão	criterion	gini
	max_depth	ilimitado
	min_samples_split	2
	splitter	best
Florestas Aleatórias	criterion	gini
	max_depth	ilimitado
	max_features	sqrt
	n_estimators	10
SVC	C	1.0
	gamma	auto
	kernel	rbf
KNN	metric	minkowski
	n_neighbors	5

Tabela 4.1: Parametros padrão

Algoritmo	Hiperparâmetro	Valor
Árvore de Decisão	criterion	[gini,entropy]
	max_depth	[10,20,30,40,50,60,70]
	splitter	[best, random]
Florestas Aleatórias	criterion	[gini,entropy]
	max_depth	[10,20,30,40,50,60,70]
	max_features	[sqrt, log2]
	n_estimators	[50, 100, 150, 200, 300]
SVC	C	[0.1, 1.0, 10, 100, 1000]
	gamma	[auto, scale]
	kernel	[linear, poly, rbf]
	degree	(1,4)
KNN	metric	[minkowski, euclidean, manhattan]
	n_neighbors	(1,15)

Tabela 4.2: Valores discretos e intervalos dos hiperparâmetros na otimização com *GridSearch*

4.7 Ferramentas e Bibliotecas

A linguagem de programação adotada foi o Python, uma das linguagens mais populares em projetos científicos de Aprendizagem Automática e ciência de dados. O Python tem uma extensiva coleção de bibliotecas fáceis de usar e com ambientes de desenvolvimento interativos. O ambiente de desenvolvimento adotado inclui a versão 3.7 do Python e tem como principais bibliotecas:

- **Scikit-Learn** [6]: biblioteca que contém implementações de algoritmos de Aprendizagem Automática e métricas de avaliação.
- **Pandas** [28]: biblioteca bastante usada para manipular e processar dados.
- **Imbalanced-learn** [24]: biblioteca que fornece ferramentas de reamostragem.
- **Numpy** [16]: biblioteca que suporta o processamento de arrays multidimensionais.
- **Matplotlib**[18]: biblioteca dedicada à visualização de dados através de gráficos como histogramas, gráficos de barras, etc.

5

Resultados

Este capítulo contém os resultados das várias experiências realizadas ao longo desta dissertação. Foi estudada a multiclassificação das 12 regiões definidas que indicam o melhor sítio para onde lançar a bola em cada jogada. Os quatro modelos escolhidos foram treinados e validados sobre os mesmos conjuntos de dados e com otimização de hiperparâmetros com Validação Cruzada. Os modelos foram treinados com a distribuição original de dados e com a distribuição equilibrada resultante da sobreamostragem dos dados, com objetivo de comparar o impacto do treino com técnica de sobre-amostragem.

Ainda neste capítulo são apresentados os resultados dos modelos desenvolvidos sobre os dados gerados pelo Sistema de Visão num conjunto de vídeos de Padel. Os vídeos foram anteriormente escolhidos para anotação e assim contêm os instantes das jogadas, a informação de que jogador bateu a bola, se a jogada foi um serviço e a jogada ideal.

5.1 Resultados

A divisão de dados descrita na secção 4.4 agrupou 707 amostras de treino e 304 amostras de teste. A otimização dos hiperparâmetros ocorreu durante a fase de treino e os parâmetros otimizados para cada modelo são:

- Árvore de Decisão
  - *criterion* : gini
  - *max_depth* : 20
  - *splitter* : best
- Florestas Aleatórias
  - *criterion* : gini
  - *max_depth* : 10
  - *max_features* : log2
  - *n_estimators* : 300
- K vizinhos mais próximos
  - *metric* : manhattan
  - *n_neighbors* : 10
- SVM
  - *C* : 100
  - *gamma* : scale
  - *kernel* : rbf

5.1.1 Matriz de Confusão e Relatório de Classificação

Os classificadores são treinados com o objetivo de associar corretamente a classe correta de cada amostra. A matriz de confusão resume a exatidão com que as classes foram previstas e em que classes o classificador fez confusão com outras classes diferentes da correta. A matriz de confusão apresentada tem, para cada modelo treinado, além da informação absoluta e relativa à quantidade de amostras de teste de cada classe, na última coluna tem a métrica da precisão e na última linha, a métrica da abrangência. No Relatório de Classificação estão novamente as métricas da precisão e abrangência com o acréscimo do suporte e do F1-score, que é uma métrica que tem em conta a abrangência e a precisão.

A matriz de confusão e relatório de classificação do classificador obtido pelo algoritmo Árvore de Decisão estão presentes na figura 5.1 e tabela 5.1, através da sua análise constata-se que o classificador tem uma exatidão de 64.8%. O classificador tem uma precisão média ponderada de 66%. As classes 0, 3, 8, 9, 11 destacam-se negativamente com uma precisão com valores na casa dos 50%, abaixo da média. A classe 0 é frequentemente confundida com classe 3 e vice-versa. As classes 0 e 3 são as classes que representam os cantos mais próximos do final do campo. Além da classe 0, o classificador confunde a classe 3 com a classe 4. Quanto à abrangência, a média ponderada do classificador é 65% e destaca-se novamente negativamente a classe 8 em que o classificador identificou corretamente apenas 40% das suas amostras.

A matriz de confusão e relatório de classificação do classificador obtido pelo algoritmo Florestas Aleatórias estão presentes na figura 5.2 e tabela 5.2, através da sua análise constata-se que o classificador tem uma exatidão de 74,0%. A precisão e abrangência média ponderada são respetivamente 75% e 74%. Comparativamente ao classificador anterior, as métricas subiram 10%. O problema da confusão entre as classes 0, 3 e 4 persiste, mas em menor quantidade. A classe com pior F1-score é a classe 8, tal como no classificador anterior. Em sentido contrário, as classes 6 e 11 estão em evidência com valores de F1-score superiores a 90%.

A matriz de confusão e relatório de classificação do classificador obtido pelo algoritmo K vizinhos mais próximos estão presentes na figura 5.3 e tabela 5.3, através da sua análise constata-se que o classificador tem uma exatidão de 72.4%. A precisão e abrangência média ponderada são respetivamente 75% e 73%. Comparativamente aos classificadores anteriores, a problemática classe 8 obteve um F1-score de 64%.

A matriz de confusão e relatório de classificação do classificador obtido pelo algoritmo SVM estão presentes na figura 5.4 e tabela 5.4, através da sua análise constata-se que o classificador tem uma exatidão de 73,0%. A precisão e abrangência média ponderada são respetivamente 74% e 73%. O classificador tem resultados interessantes na medida em que todas as classes têm pelo menos 67% de F1-score, ou seja, não existe grande discrepância na classificação entre classes, situação comum nos classificadores anteriores.

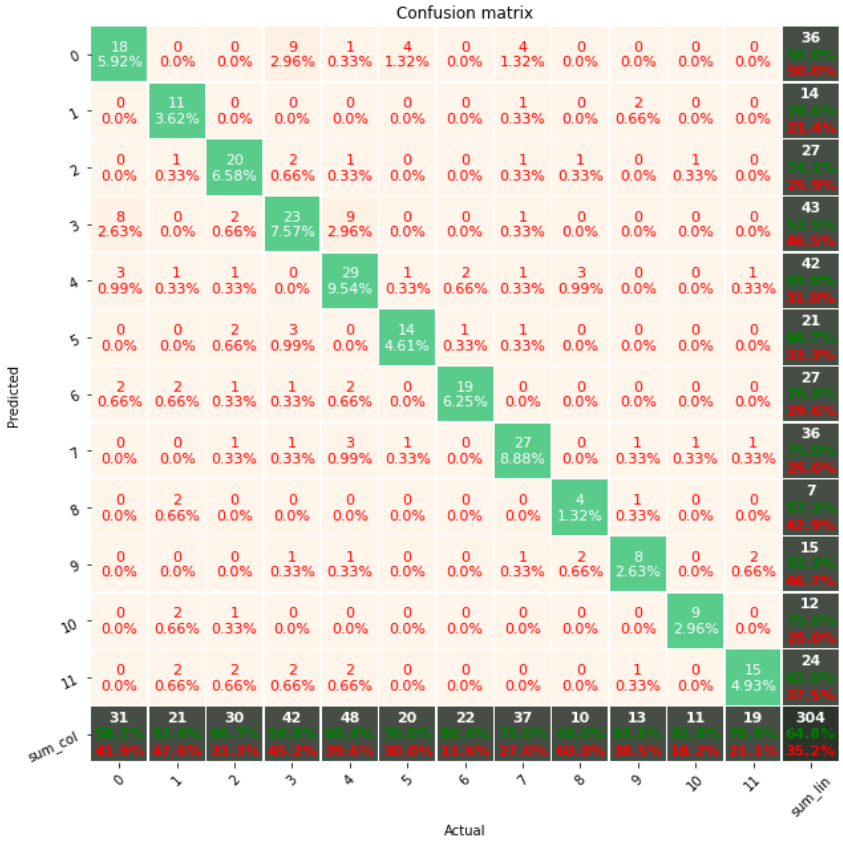


Figura 5.1: Matriz de confusão do classificador Árvore de Decisão

Classe	Precisão	Abrangência	F1-score	Suporte
0	0.50	0.58	0.54	31
1	0.79	0.52	0.63	21
2	0.74	0.67	0.70	30
3	0.53	0.55	0.54	42
4	0.69	0.60	0.64	48
5	0.67	0.70	0.68	20
6	0.70	0.86	0.78	22
7	0.75	0.73	0.74	37
8	0.57	0.40	0.47	10
9	0.53	0.62	0.57	13
10	0.75	0.82	0.78	11
11	0.62	0.79	0.70	19
exatidão			0.65	304
macro avg	0.65	0.65	0.65	304
weighted avg	0.66	0.65	0.65	304

Tabela 5.1: Relatório de classificação do classificador Árvore de Decisão

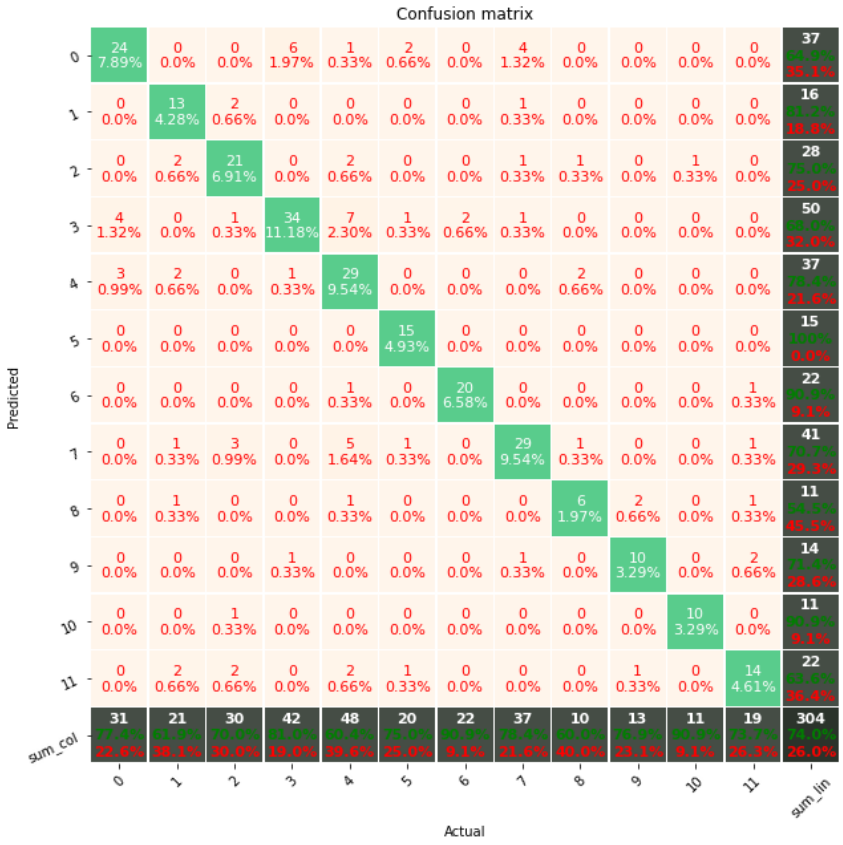


Figura 5.2: Matriz de confusão do classificador Florestas Aleatórias

Classe	Precisão	Abrangência	F1-score	Suporte
0	0.65	0.77	0.71	31
1	0.81	0.62	0.70	21
2	0.75	0.70	0.72	30
3	0.68	0.81	0.74	42
4	0.78	0.60	0.68	48
5	1.00	0.75	0.86	20
6	0.91	0.91	0.91	22
7	0.71	0.78	0.74	37
8	0.55	0.60	0.57	10
9	0.71	0.77	0.74	13
10	0.91	0.91	0.91	11
11	0.64	0.74	0.68	19
exatidão			0.74	304
macro avg	0.76	0.75	0.75	304
weighted avg	0.75	0.74	0.74	304

Tabela 5.2: Relatório de classificação do classificador Florestas Aleatórias

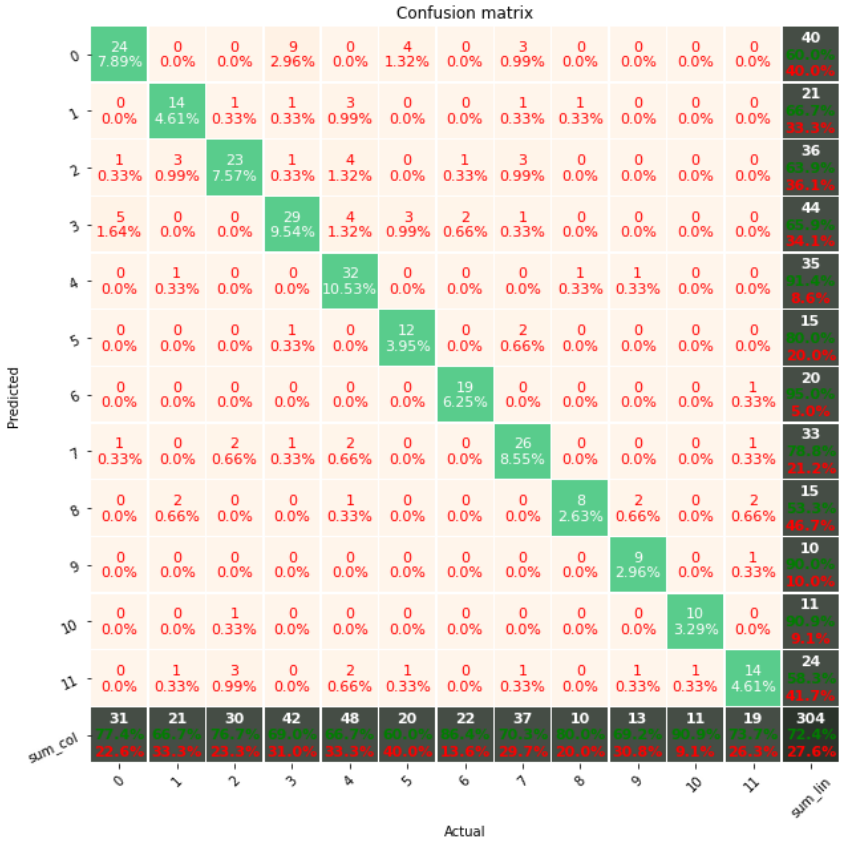


Figura 5.3: Matriz de confusão do classificador K vizinhos mais próximos

Classe	Precisão	Abrangência	F1-score	Suporte
0	0.60	0.77	0.68	31
1	0.67	0.67	0.67	21
2	0.64	0.77	0.70	30
3	0.66	0.69	0.67	42
4	0.91	0.67	0.77	48
5	0.80	0.60	0.69	20
6	0.95	0.86	0.90	22
7	0.79	0.70	0.74	37
8	0.53	0.80	0.64	10
9	0.90	0.69	0.78	13
10	0.91	0.91	0.91	11
11	0.58	0.74	0.65	19
exatidão			0.72	304
macro avg	0.75	0.74	0.73	304
weighted avg	0.75	0.72	0.73	304

Tabela 5.3: Relatório de classificação do classificador K vizinhos mais próximos com k = 10

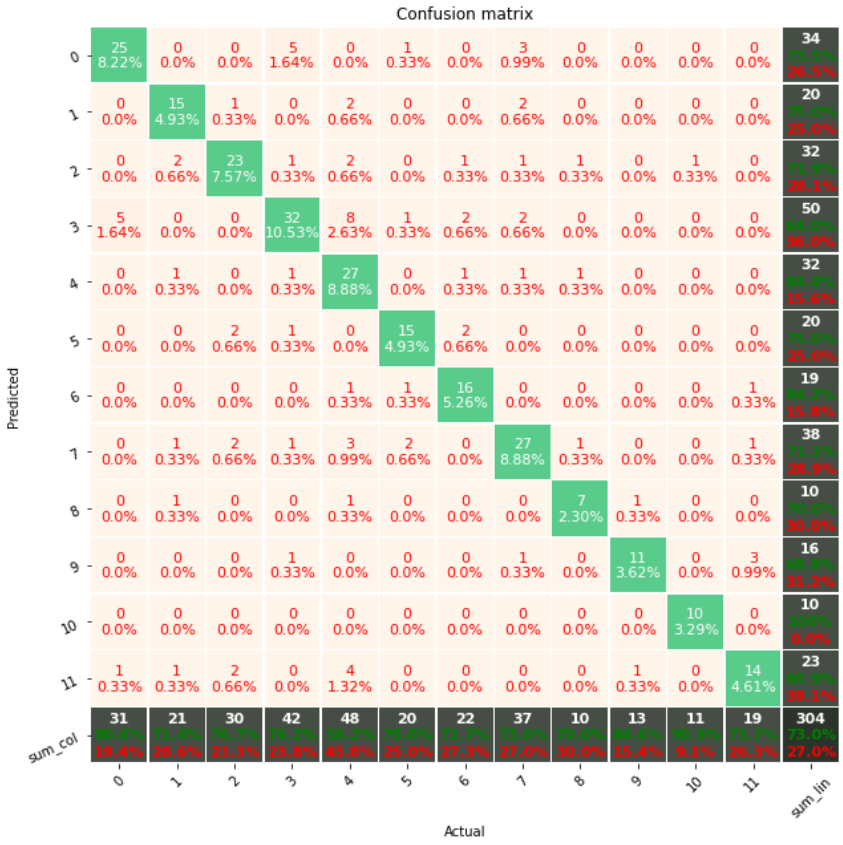


Figura 5.4: Matriz de confusão do classificador SVM

Classe	Precisão	Abrangência	F1-score	Suporte
0	0.74	0.81	0.77	31
1	0.75	0.71	0.73	21
2	0.72	0.77	0.74	30
3	0.64	0.76	0.70	42
4	0.84	0.56	0.68	48
5	0.75	0.75	0.75	20
6	0.84	0.73	0.78	22
7	0.71	0.73	0.72	37
8	0.70	0.70	0.70	10
9	0.69	0.85	0.76	13
10	1.00	0.91	0.95	11
11	0.61	0.74	0.67	19
exatidão			0.73	304
macro avg	0.75	0.75	0.75	304
weighted avg	0.74	0.73	0.73	304

Tabela 5.4: Relatório de classificação do classificador SVM

5.1.2 Curvas AUC-ROC

Os gráficos abaixo apresentam as curvas ROC relativas a cada classe para cada um dos classificadores. As curvas ROC são influenciadas pela quantidade de amostras de teste usadas, assim sendo, as classes com menor número de amostras têm menos pontos nas curvas correspondentes. A AUC, presente na legenda, indica que quanto maior o seu valor, melhor a capacidade do modelo distinguir a classe em consideração das restantes.

O classificador Árvore de Decisão apresenta valores de AUC na casa dos 70 e 80%. Os restantes classificadores apresentam valores de AUC semelhantes entre si no intervalo entre os 88 e 100%. Nos gráficos é possível comparar quais os classificadores que distinguem melhor certas classes, por exemplo, na classe 4, a mais numerosa, os melhores classificadores são os obtidos pelos algoritmos K vizinhos mais próximos e Florestas Aleatórias por apresentarem o valor de AUC igual a 90%. Na classe 7, os melhores classificadores são os obtidos pelos algoritmos Florestas Aleatórias e SVM com valor de AUC igual a 92%. Em suma, os classificadores testados conseguem distinguir todas as classes com sucesso.

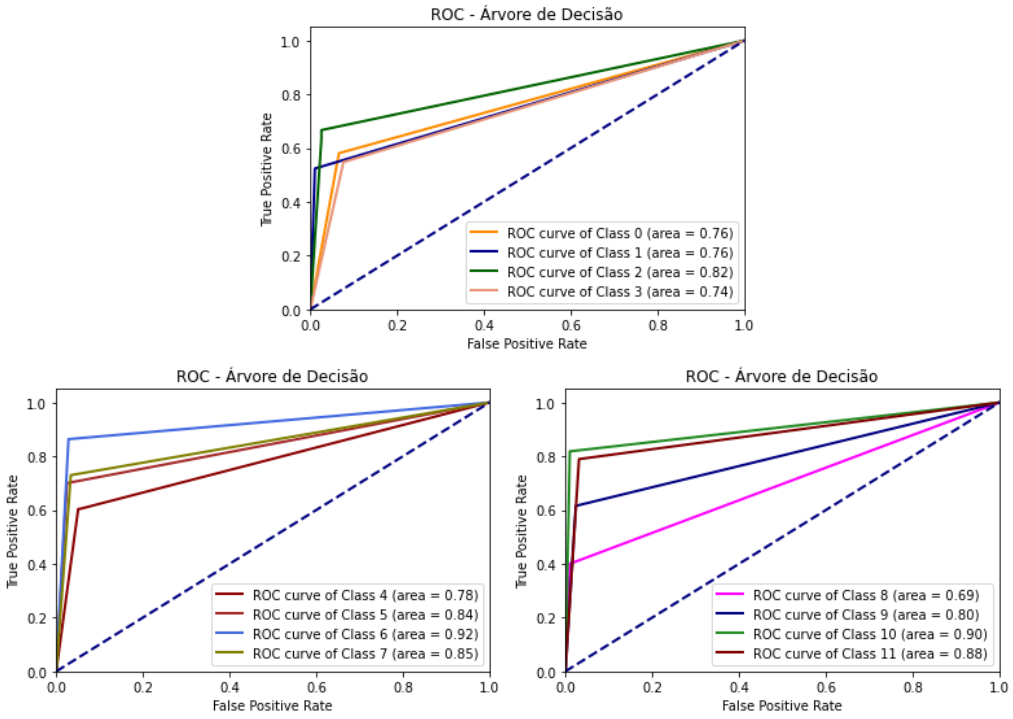


Figura 5.5: Curva AUC-ROC do classificador Árvore de Decisão

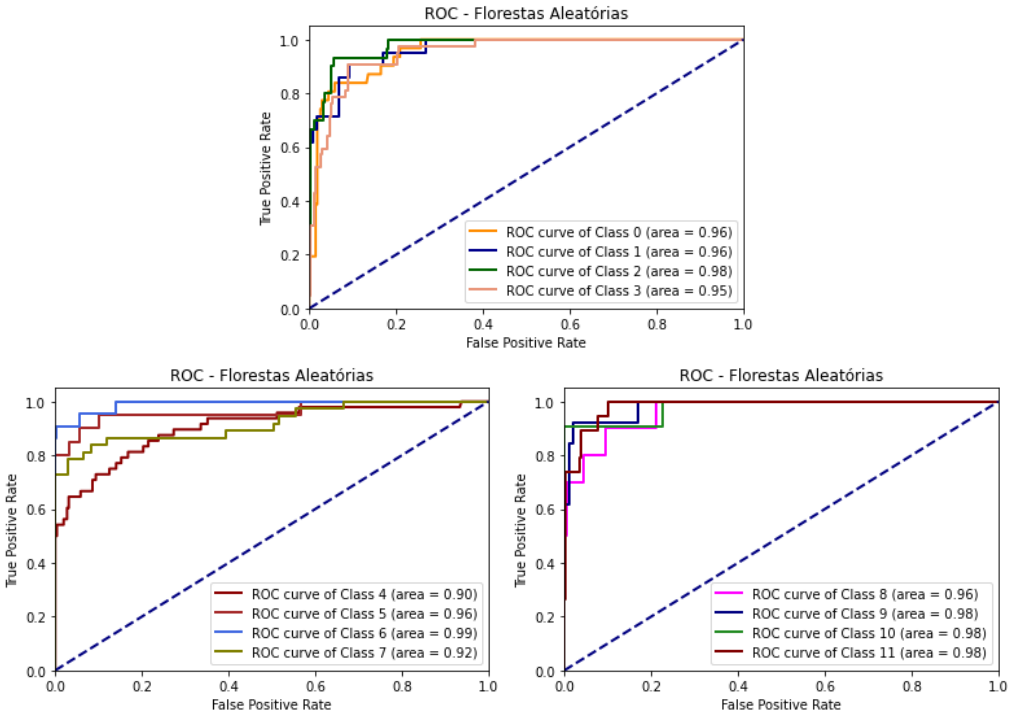


Figura 5.6: Curva AUC-ROC do classificador Florestas Aleatórias

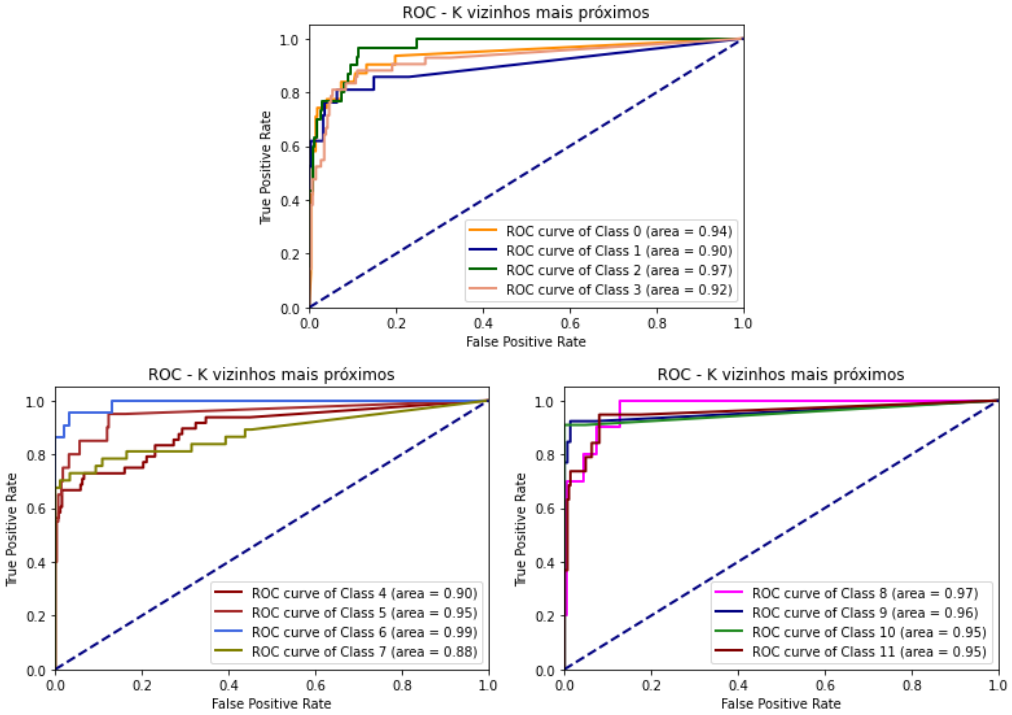


Figura 5.7: Curva AUC-ROC do classificador K vizinhos mais próximos

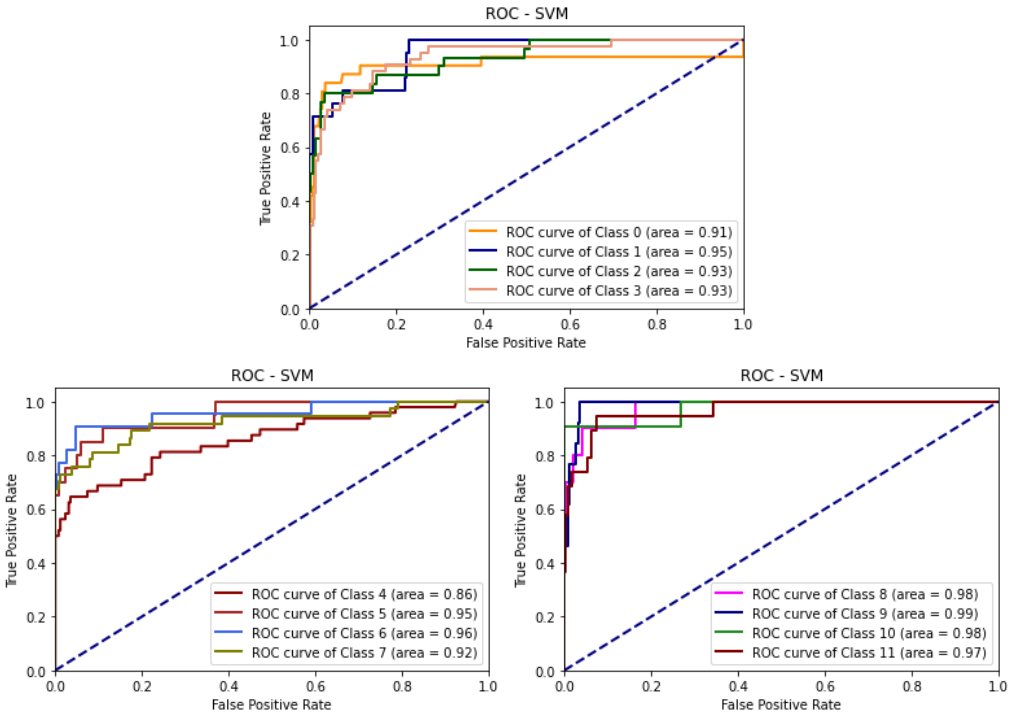


Figura 5.8: Curva AUC-ROC do classificador SVM

5.2 Dados sobreamostrados

Nesta secção são apresentadas as métricas dos relatórios de classificação para cada um dos classificadores treinados com sobre-amostragem dos dados de treino. Os hiperparâmetros foram otimizados com a técnica de *GridSearch* e como expetável diferem dos classificadores treinados com a distribuição original de dados. Os hiperparâmetros otimizados para cada modelo são:

- Árvore de Decisão
  - *criterion* : gini
  - *max_depth* : 10
  - *splitter* : best
- Florestas Aleatórias
  - *criterion* : gini
  - *max_depth* : 70
  - *max_features* : log2
  - *n_estimators* : 200
- K vizinhos mais próximos
  - *metric* : euclidean
  - *n_neighbors* : 2
- SVM
  - *C* : 1000
  - *gamma* : scale
  - *kernel* : rbf

Através da comparação dos relatórios de classificação entre os classificadores originais e sobre-amostrados verifica-se que os classificadores obtidos pelos algoritmos K vizinhos

mais próximos e SVM apresentam piores métricas com a distribuição equilibrada de dados resultante da sobre-amostragem. Os dois classificadores obtidos pelos algoritmos Árvore de Decisão e Florestas Aleatórias apresentam melhores resultados com a distribuição equilibrada resultante da sobre-amostragem.

O classificador Árvore de Decisão tem uma exatidão de 68%, subindo 3 valores percentuais em relação ao classificador homólogo. As métricas de precisão e abrangência média ponderada também sobem para 70% e 68%. A Classe 8 que anteriormente tinha um F1-score de 47% passou para 67%. Em sentido contrário a Classe 1 que era 63% e passou para 59%. As restantes classes em média subiram os valores de F1-score.

O classificador Florestas Aleatórias tem uma exatidão de 74%, igual ao classificador homólogo. A diferença entre o classificador atual e o homólogo está presente na métrica de precisão média ponderada que sobe de 75% para 76%. A problemática Classe 8 continuou com mesmo valor de 57% de F1-score.

Classe	Precisão	Abrangência	F1-score	Suporte
0	0.62	0.65	0.63	31
1	0.77	0.48	0.59	21
2	0.57	0.67	0.62	30
3	0.58	0.67	0.62	42
4	0.76	0.60	0.67	48
5	0.68	0.65	0.67	20
6	0.72	0.95	0.82	22
7	0.79	0.70	0.74	37
8	0.64	0.70	0.67	10
9	0.83	0.77	0.80	13
10	0.91	0.91	0.91	11
11	0.61	0.74	0.67	19
exatidão			0.68	304
macro avg	0.71	0.71	0.70	304
weighted avg	0.70	0.68	0.68	304

Tabela 5.5: Relatório de classificação do classificador Árvore de Decisão com dados sobre-amostrados

Classe	Precisão	Abrangência	F1-score	Suporte
0	0.68	0.74	0.71	31
1	0.59	0.62	0.60	21
2	0.65	0.80	0.72	30
3	0.65	0.74	0.69	42
4	0.91	0.60	0.72	48
5	0.80	0.80	0.80	20
6	0.87	0.91	0.89	22
7	0.82	0.73	0.77	37
8	0.55	0.60	0.57	10
9	0.91	0.77	0.83	13
10	1.00	0.91	0.95	11
11	0.70	0.84	0.76	19
exatidão			0.74	304
macro avg	0.76	0.76	0.75	304
weighted avg	0.76	0.74	0.74	304

Tabela 5.6: Relatório de classificação do classificador Florestas Aleatórias com dados sobre-amostrados

Classe	Precisão	Abrangência	F1-score	Suporte
0	0.69	0.77	0.73	31
1	0.52	0.67	0.58	21
2	0.65	0.67	0.66	30
3	0.72	0.69	0.71	42
4	0.91	0.65	0.76	48
5	0.68	0.65	0.67	20
6	0.76	0.86	0.81	22
7	0.72	0.70	0.71	37
8	0.47	0.80	0.59	10
9	0.80	0.62	0.70	13
10	0.91	0.91	0.91	11
11	0.63	0.63	0.63	19
exatidão			0.70	304
macro avg	0.71	0.72	0.70	304
weighted avg	0.72	0.70	0.71	304

Tabela 5.7: Relatório de classificação do classificador K vizinhos mais próximos com dados sobre-amostrados com k = 2

5.3 Resultados sobre vídeo de Padel

Até esta secção, os classificadores foram testados sobre dados anotados pelos especialistas. Nesta secção de resultados pretende-se comparar os resultados dos classificadores trabalhados em três vídeos de Padel que têm a posição dos jogadores automaticamente detetada. Os três vídeos escolhidos foram anteriormente anotados porque é necessário saber os instantes em que houve jogadas, quem as praticou, se estas foram um serviço e a classe atribuída. A deteção da posição dos jogadores é realizada automaticamente através

Classe	Precisão	Abrangência	F1-score	Suporte
0	0.66	0.74	0.70	31
1	0.74	0.67	0.70	21
2	0.59	0.77	0.67	30
3	0.68	0.67	0.67	42
4	0.83	0.60	0.70	48
5	0.57	0.60	0.59	20
6	0.84	0.73	0.78	22
7	0.68	0.70	0.69	37
8	0.45	0.50	0.48	10
9	0.64	0.69	0.67	13
10	1.00	0.91	0.95	11
11	0.64	0.74	0.68	19
exatidão			0.69	304
macro avg	0.69	0.69	0.69	304
weighted avg	0.70	0.69	0.69	304

Tabela 5.8: Relatório de classificação do classificador SVM com dados sobre-amostrados

do sistema montado e descrito na secção 3.3. As posições dos jogadores automaticamente detetadas diferem das anotadas, o que provoca interesse no desempenho dos classificadores nestas circunstâncias. A seleção de três vídeos foi efetuada de forma aleatória entre os anotados e com o conhecimento que os vídeos não foram usados para o treino do detetor de objetos.

Os classificadores testados foram os classificadores com as melhores métricas, ou seja, classificador de Árvore de Decisão treinado com os dados sobre-amostrados, classificador Florestas Aleatórias treinado também com os dados sobre-amostrados, classificador K vizinhos mais próximos e classificador SVM treinados com os dados originais.

Os resultados dos classificadores sobre os dados dos jogadores automaticamente detetados são bastante positivos, o que conclui que as anotações realizadas foram um bom conjunto de treino para os classificadores. As classes 5, 6, 9, 10 têm poucas amostras, o que indica que as métricas para estas classes apresentadas nas tabelas não são as mais representativas.

O relatório de classificação do classificador treinado pelo algoritmo Árvore de Decisão está presente na tabela 5.9 . O classificador obteve uma exatidão de 80%. Os valores de F1-score estão na maioria entre o intervalo dos 70% aos 93%. A média ponderada do F1-score é 81%.

O relatório de classificação do classificador treinado pelo algoritmo Florestas Aleatórias está presente na tabela 5.10 . O classificador obteve uma exatidão de 88%. Os valores de F1-score também estão na maioria entre o intervalo dos 70% aos 95%. A média ponderada do F1-score é 88%.

O relatório de classificação do classificador treinado pelo algoritmo K vizinhos mais próximos está presente na tabela 5.11. O classificador obteve uma exatidão de 89%. Os valores de F1-score excetuando as classes minoritárias estão no intervalo entre 74% e

100%. A média ponderada do F1-score é 88%.

O relatório de classificação do classificador treinado pelo algoritmo SVM está presente na tabela 5.12 . O classificador obteve uma exatidão de 86%. Os valores de F1-score estão na maioria entre o intervalo de 60% a 97%. A média ponderada do F1-score é 86%.

Os resultados obtidos são interpretados como a validação dos classificadores desenvolvidos com os dados gerados pelo Sistema de Visão. As métricas mais representativas dos classificadores são as da secção anterior por serem mais numerosas e representativas das diferentes classes.

Classe	Precisão	Abrangência	F1-score	Suporte
0	0.93	0.93	0.93	28
1	0.58	0.88	0.70	8
2	0.75	0.79	0.77	19
3	0.85	0.74	0.79	23
4	0.85	0.69	0.76	16
5	0.29	0.40	0.33	5
6	0.50	0.50	0.50	2
7	0.90	0.82	0.86	33
8	0.82	0.82	0.82	11
9	1.00	1.00	1.00	2
10	1.00	1.00	1.00	1
11	0.70	0.88	0.78	8
exatidão			0.80	156
macro avg	0.76	0.79	0.77	156
weighted avg	0.82	0.80	0.81	156

Tabela 5.9: Relatório de classificação do classificador Árvore de Decisão sobre os dados gerados pelo detetor de objetos

Classe	Precisão	Abrangência	F1-score	Suporte
0	0.90	0.96	0.93	28
1	0.64	0.88	0.74	8
2	0.76	0.84	0.80	19
3	0.92	0.96	0.94	23
4	0.92	0.75	0.83	16
5	0.67	0.40	0.50	5
6	1.00	0.50	0.67	2
7	0.97	0.94	0.95	33
8	1.00	0.82	0.90	11
9	1.00	1.00	1.00	2
10	1.00	1.00	1.00	1
11	0.78	0.88	0.82	8
exatidão			0.88	156
macro avg	0.88	0.83	0.84	156
weighted avg	0.89	0.88	0.88	156

Tabela 5.10: Relatório de classificação do classificador Florestas Aleatórias sobre os dados gerados pelo detetor de objetos

Classe	Precisão	Abrangência	F1-score	Suporte
0	0.93	1.00	0.97	28
1	0.64	0.88	0.74	8
2	0.85	0.89	0.87	19
3	0.95	0.87	0.91	23
4	0.88	0.88	0.88	16
5	1.00	0.60	0.75	5
6	0.00	0.00	0.00	2
7	0.91	0.91	0.91	33
8	1.00	1.00	1.00	11
9	1.00	0.50	0.67	2
10	0.00	0.00	0.00	1
11	0.89	1.00	0.94	8
exatidão			0.89	156
macro avg	0.75	0.71	0.72	156
weighted avg	0.89	0.89	0.88	156

Tabela 5.11: Relatório de classificação do classificador K vizinhos mais próximos o sobre os dados gerados pelo detetor de objetos

Classe	Precisão	Abrangência	F1-score	Suporte
0	0.93	1.00	0.97	28
1	0.50	0.75	0.60	8
2	0.84	0.84	0.84	19
3	0.81	0.91	0.86	23
4	0.87	0.81	0.84	16
5	1.00	0.40	0.57	5
6	0.00	0.00	0.00	2
7	1.00	0.88	0.94	33
8	1.00	0.82	0.90	11
9	0.67	1.00	0.80	2
10	0.50	1.00	0.67	1
11	0.78	0.88	0.82	8
exatidão			0.86	156
macro avg	0.74	0.77	0.73	156
weighted avg	0.87	0.86	0.86	156

Tabela 5.12: Relatório de classificação do classificador SVM sobre os dados gerados pelo detetor de objetos

6

Conclusões e Trabalho Futuro

6.1 Conclusões

O trabalho desenvolvido desta dissertação provou, de modo geral, ser possível utilizar técnicas de deteção, *tracking* e Aprendizagem Automática no Padel. As técnicas estudadas já foram aplicadas em desportos tanto coletivos como individuais, no entanto, era algo imprevisível o comportamento das mesmas no Padel.

Relembrando os objetivos propostos, a componente de Sistema de Visão conseguiu detetar e monitorizar cada jogador individualmente. O paradigma adotado nesta componente foi o *tracking-by-detectio*n que aplica primeiro um algoritmo de deteção de objetos e a seguir um algoritmo de *tracking*. Os resultados da deteção, detalhados na secção 3.3, mostraram ser excelentes com o modelo YOLO a obter um mAP igual a 99.82%. Sobre o *tracking*, a aplicação do Deep SORT resultou em apenas uma troca de identidade que resulta num IDF1 igual a 96.8%.

Ainda nesta componente foi realizado o mapeamento do campo que permitiu mostrar a posição dos jogadores no modelo do campo 2D. Por fim, foi realizado um pequeno estudo de como detetar momentos de jogada automaticamente através dos mínimos da distância entre os jogadores e a bola. A proposta apresentada resultou num elevado número de falsos positivos e por isso não foi levada adiante.

A componente do Treinador Virtual foi desenvolvida com o intuito de alcançar o último objetivo proposto, sugerir aos jogadores uma melhor estratégia nos momentos em que jogam a bola. Nesta componente foram percorridas as várias etapas de um problema de Aprendizagem Automática, desde a recolha e anotação de dados até à extração dos resultados.

A recolha e anotação dos dados resultou como contribuição um conjunto de dados com a várias jogadas ideais no contexto do Padel. O conjunto de dados foi anotado por 2 jogadores experientes de modo mutuamente exclusivo visando chegar a um conjunto de dados o mais rapidamente possível. No entanto, idealmente, o conjunto de dados seria anotado apenas por um especialista, visto que apontar uma jogada ideal em várias situações de jogo é um exercício subjetivo.

O conjunto de dados recolhido é algo desequilibrado, visto que as mecânicas do Padel tendem a favorecer lançar a bola para as zonas mais próximas das paredes de vidro. Assim sendo, os modelos foram treinados com os dados originais e com a sobre-amostragem dos dados. Os classificadores ótimos obtidos apresentaram resultados bastante semelhantes com métricas de exatidão, precisão, abrangência e F1-score na casa dos 70%. A escolha da adoção de um classificador como Treinador Virtual é o modelo treinado com sobre-amostragem dos dados e com o algoritmo Florestas Aleatórias.

A interpretação dos resultados do Treinador Virtual deve ser realizada como uma sugestão de melhor jogada e não como uma certeza que a jogada indicada é a melhor. Esta situação é justificada por o Padel ser por si um jogo subjetivo e as métricas obtidas estarem por valores à volta dos 75%.

Na última etapa do trabalho, os dados da posição dos jogadores gerados pelo Sistema de Visão foram testados pelo Treinador Virtual e apresentaram bons resultados que validam a utilização conjunta das duas componentes.

6.2 Trabalho Futuro

Como enunciado ao longo desta dissertação a principal componente desenvolvida é o Treinador Virtual. O Sistema de Visão foi importante para estudar que atributos anotar e como fonte de dados para a aplicação móvel. No entanto, sobre o sistema de visão só foi estudado o paradigma *tracking-by-detection*, que é um paradigma de duas fases, que aplica um algoritmo de deteção de objetos e depois um algoritmo de *tracking*. Ainda sobre este paradigma foi testado apenas um algoritmo para cada modelo, o YOLO e depois o Deep SORT. A escolha e teste de outros algoritmos no paradigma enunciado e aplicação de um paradigma de uma fase seria enriquecedor.

A secção 3.2 referente ao mapeamento do campo menciona que os 4 pontos que representam os limites do campo na imagem foram extraídos manualmente, porém este passo pode ser efetuado automaticamente recorrendo a técnicas de processamento de imagem que consigam extrair os cantos do campo.

O Sistema de Visão foi preparado para receber sequências de vídeo capturadas por apenas uma câmara, no entanto, esta escolha limitou os dados extraídos, já que só com uma câmara é impossível inferir informação 3D presente no jogo. Esta informação seria importante no mapeamento dos elementos físicos presentes no campo tais como a rede e as paredes de vidro. Também, e como explicado na secção 3.5, a localização dos jogadores seria mais uniforme, visto que com uma câmara o detetor de objetos "vê" uma dupla de jogadores de costas e a adversária de frente. A colocação de mais câmaras seria benéfica para extração de informação com mais exatidão.

A componente de Treinador Virtual representa a aplicação de Aprendizagem Automática. Como em qualquer problema de AA, a disponibilidade e processamento dos dados são requisitos cruciais para o sucesso. A disponibilidade de dados no início desta dissertação era nula, situação que obrigou à construção de um conjunto de dados. Como

referido na secção anterior a anotação dos dados foi realizada por dois especialistas, no entanto, idealmente a anotação seria realizada apenas por um porque a escolha da melhor jogada é um conceito subjetivo. A proposta de trabalho futuro é que a recolha e anotação do conjunto de dados seja realizada por apenas um especialista e seja conduzido um estudo/questionário que indique de forma geral a abordagem do especialista, isto é, averiguar se o especialista tem uma abordagem mais conservadora ou atacante em relação ao jogo e assim enquadrar o treinador virtual desenvolvido para o nível do utilizador.

Os dados anotados têm como atributos a posição de cada jogador e se a jogada foi um serviço. O trabalho desenvolvido convida à extração de novos atributos a partir dos dados existentes como, por exemplo, a velocidade do jogador no momento da jogada. Como também à integração de atributos relacionados com a bola, por exemplo, velocidade da bola, distância percorrida, número de toques ocorridos no ponto, se a bola tocou no vidro, entre outros.

Como enunciado ao longo da dissertação, a extração da posição de cada jogador é automática, contudo as informações sobre em que momentos houve uma jogada, se a jogada foi um serviço e que jogador a realizou não são extraídos automaticamente. O estudo sobre o reconhecimento de ação pode ajudar a recolher as informações enunciadas.

Por fim, os dados espaço-temporais gerados pelo Sistema de Visão e Treinador Virtual convidam ao uso de várias Visualizações de Dados pertinentes para os utilizadores da futura aplicação.

Bibliografia

[1] *Australia Open*. Accessed: 2020-02-21. URL: https://ausopen.com/ (ver p. 6).

[2] A. Bewley et al. "Simple Online and Realtime Tracking". Em: *2016 IEEE International Conference on Image Processing (ICIP)* (set. de 2016). arXiv: 1602.00763, pp. 3464–3468. DOI: 10.1109/ICIP.2016.7533003. URL: http://arxiv.org/abs/1602.00763 (acedido em 15/08/2021) (ver p. 18).

[3] A. Bochkovskiy, C.-Y. Wang e H.-Y. M. Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection". Em: *arXiv:2004.10934 [cs, eess]* (abr. de 2020). arXiv: 2004.10934. URL: http://arxiv.org/abs/2004.10934 (acedido em 30/09/2021) (ver p. 18).

[4] G. Bradski. "The OpenCV Library". Em: *Dr. Dobb's Journal of Software Tools* (2000) (ver pp. 22, 28).

[5] M. D. Breitenstein et al. "Online Multiperson Tracking-by-Detection from a Single, Uncalibrated Camera". Em: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.9 (set. de 2011). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1820–1833. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2010.232 (ver p. 29).

[6] L. Buitinck et al. "API design for machine learning software: experiences from the scikit-learn project". Em: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122 (ver p. 46).

[7] M. Buric, M. Ivasic-Kos e M. Pobar. "Player Tracking in Sports Videos". Em: *2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. ISSN: 2330-2186. Dez. de 2019, pp. 334–340. DOI: 10.1109/CloudCom.2019.00058 (ver p. 23).

[8] N. V. Chawla et al. "SMOTE: Synthetic Minority Over-sampling Technique". Em: *Journal of Artificial Intelligence Research* 16 (jun. de 2002). arXiv: 1106.1813, pp. 321–357. ISSN: 1076-9757. DOI: 10.1613/jair.953. URL: http://arxiv.org/abs/1106.1813 (acedido em 03/09/2021) (ver p. 43).

[9] M. Cordts et al. “The Cityscapes Dataset for Semantic Urban Scene Understanding”. Em: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (ver p. 15).

[10] L. Draschkowitz, C. Draschkowitz e H. Hlavacs. “Using Video Analysis and Machine Learning for Predicting Shot Success in Table Tennis”. en. Em: *EAI Endorsed Transactions on Creative Technologies* 2.5 (out. de 2015), p. 150096. ISSN: 2409-9708. DOI: 10.4108/eai.20-10-2015.150096. URL: http://eudl.eu/doi/10.4108/eai.20-10-2015.150096 (acedido em 04/07/2021) (ver p. 25).

[11] M. A. H. Eibe Frank e I. H. Witten. “The Weka Workbench”. Em: *Data Mining: Practical Machine Learning Tools and Techniques*. 2016 (ver p. 25).

[12] *Federação Portuguesa de Padel : REGRAS DE JOGO*. Accessed: 2020-02-21. URL: https://www.fppadel.pt/document/6A5020B6-FD90-4684-B5A6-F4382163B0C (ver p. 5).

[13] G. M. Foody e A. Mathur. “Toward intelligent training of supervised image classifications: directing training data acquisition for SVM classification”. Em: *Remote Sensing of Environment* 93.1 (2004), pp. 107–117. ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2004.06.017. URL: https://www.sciencedirect.com/science/article/pii/S003442570400207X (ver p. 7).

[14] R. Girshick. “Fast R-CNN”. en. Em: (abr. de 2015). URL: https://arxiv.org/abs/1504.08083v2 (acedido em 24/05/2021) (ver p. 16).

[15] R. Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. Em: *arXiv:1311.2524 [cs]* (out. de 2014). arXiv: 1311.2524 version: 5. URL: http://arxiv.org/abs/1311.2524 (acedido em 19/07/2021) (ver p. 16).

[16] C. R. Harris et al. “Array programming with NumPy”. Em: *Nature* 585.7825 (set. de 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2 (ver p. 46).

[17] Y.-C. Huang et al. “TrackNet: A Deep Learning Network for Tracking High-speed and Tiny Objects in Sports Applications”. Em: *arXiv:1907.03698 [cs, stat]* (jul. de 2019). arXiv: 1907.03698 version: 1. URL: http://arxiv.org/abs/1907.03698 (acedido em 09/02/2022) (ver p. 30).

[18] J. D. Hunter. “Matplotlib: A 2D graphics environment”. Em: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55 (ver p. 46).

[19] M. Ivašić-Kos, K. Host e M. Pobar. “Application of Deep Learning Methods for Detection and Tracking of Players”. Em: fev. de 2021. ISBN: 978-1-83962-374-5. DOI: 10.5772/intechopen.96308 (ver p. 23).

[20] G. H. John e P. Langley. "Estimating Continuous Distributions in Bayesian Classifiers". Em: *Eleventh Conference on Uncertainty in Artificial Intelligence*. San Mateo: Morgan Kaufmann, 1995, pp. 338–345 (ver p. 25).

[21] S. Kalmegh. "Analysis of WEKA Data Mining Algorithm REPTree, Simple Cart and RandomTree for Classification of Indian News". en. Em: 2.2 (), p. 9 (ver p. 25).

[22] R. Keim. *How to Train a Basic Perceptron Neural Network*. Accessed: 2020-02-22. 2019. URL: https://www.allaboutcircuits.com/technical-articles/how-to-train-a-basic-perceptron-neural-network/ (ver p. 12).

[23] R. Kohavi. "The Power of Decision Tables". Em: *8th European Conference on Machine Learning*. Springer, 1995, pp. 174–189 (ver p. 25).

[24] G. Lemaître, F. Nogueira e C. K. Aridas. "Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning". Em: *Journal of Machine Learning Research* 18.17 (2017), pp. 1–5. URL: http://jmlr.org/papers/v18/16-365 (ver p. 46).

[25] T.-Y. Lin et al. "Microsoft COCO: Common Objects in Context". en. Em: *Computer Vision – ECCV 2014*. Ed. por D. Fleet et al. Vol. 8693. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pp. 740–755. ISBN: 978-3-319-10601-4 978-3-319-10602-1. DOI: 10.1007/978-3-319-10602-1_48. URL: http://link.springer.com/10.1007/978-3-319-10602-1_48 (acedido em 04/08/2021) (ver pp. 15, 24).

[26] W. Liu et al. "SSD: Single Shot MultiBox Detector". Em: *arXiv:1512.02325 [cs]* 9905 (2016). arXiv: 1512.02325, pp. 21–37. DOI: 10.1007/978-3-319-46448-0_2. URL: http://arxiv.org/abs/1512.02325 (acedido em 14/06/2021) (ver p. 16).

[27] *mAP (mean Average Precision) for Object Detection | by Jonathan Hui | Medium*. URL: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173 (acedido em 02/09/2021) (ver pp. 20, 21).

[28] W. McKinney et al. "Data structures for statistical computing in python". Em: *Proceedings of the 9th Python in Science Conference*. Vol. 445. Austin, TX. 2010, pp. 51–56 (ver p. 46).

[29] S. Mora. "Computer Vision and Machine Learning for In-Play Tennis Analysis: Framework, Algorithms and Implementation". Tese de doutoramento. Imperial College London, 2017 (ver pp. 12, 22).

[30] M. S. Nixon e A. S. Aguado. "5 - High-level feature extraction: fixed shape matching". en. Em: *Feature Extraction and Image Processing for Computer Vision (Fourth Edition)*. Ed. por M. S. Nixon e A. S. Aguado. Academic Press, jan. de 2020, pp. 223–290. ISBN: 978-0-12-814976-8. DOI: 10.1016/B978-0-12-814976-8.00005-1. URL: https://www.sciencedirect.com/science/article/pii/B9780128149768000051 (acedido em 12/04/2021) (ver p. 22).

[31] N. Owens, C. Harris e C. Stennett. "Hawk-eye tennis system". Em: *2003 International Conference on Visual Information Engineering VIE 2003*. ISSN: 0537-9989. Jul. de 2003, pp. 182–185. DOI: 10.1049/cp:20030517 (ver p. 1).

[32] J. Platt. "Fast Training of Support Vector Machines using Sequential Minimal Optimization". Em: *Advances in Kernel Methods - Support Vector Learning*. Ed. por B. Schoelkopf, C. Burges e A. Smola. MIT Press, 1998. URL: http://research.microsoft.com/%5C~jplatt/smo.html (ver p. 25).

[33] J. Redmon e A. Farhadi. "YOLO9000: Better, Faster, Stronger". Em: *arXiv:1612.08242 [cs]* (dez. de 2016). arXiv: 1612.08242. URL: http://arxiv.org/abs/1612.08242 (acedido em 24/05/2021) (ver p. 17).

[34] J. Redmon e A. Farhadi. "YOLOv3: An Incremental Improvement". Em: *arXiv:1804.02767 [cs]* (abr. de 2018). arXiv: 1804.02767. URL: http://arxiv.org/abs/1804.02767 (acedido em 24/05/2021) (ver p. 18).

[35] J. Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". Em: *arXiv:1506.02640 [cs]* (mai. de 2016). arXiv: 1506.02640 version: 5. URL: http://arxiv.org/abs/1506.02640 (acedido em 17/09/2021) (ver p. 16).

[36] G. V. Research. "Sports Analytics Market Size, Share, Trends Analysis Report By Component (Software, Service), By Analysis Type (On-field, Off-field), By Sports (Football, Cricket, Basketball, Baseball), And Segment Forecasts, 2021 - 2028". Em: (Apr 2021) (ver p. 2).

[37] E. Ristani et al. "Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking". Em: *arXiv:1609.01775 [cs]* (set. de 2016). arXiv: 1609.01775. URL: http://arxiv.org/abs/1609.01775 (acedido em 30/09/2021) (ver p. 22).

[38] O. Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". Em: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y (ver p. 15).

[39] H. Su, J. Deng e L. Fei-Fei. "Crowdsourcing Annotations for Visual Object Detection". en. Em: (), p. 7 (ver p. 29).

[40] C. Szegedy et al. "Going Deeper with Convolutions". en. Em: (set. de 2014). URL: https://arxiv.org/abs/1409.4842v1 (acedido em 24/08/2021) (ver p. 16).

[41] *What is padel?* Accessed: 2020-02-21. URL: https://www.usaplayspadel.com/what-is-padel/ (ver pp. 1, 6).

[42] *Why is padel so popular?* Accessed: 2020-02-21. URL: https://www.tenniskit24.com/padel/why-is-padel-so-popular/ (ver p. 1).

[43] N. Wojke, A. Bewley e D. Paulus. "Simple Online and Realtime Tracking with a Deep Association Metric". Em: *arXiv:1703.07402 [cs]* (mar. de 2017). arXiv: 1703.07402. URL: http://arxiv.org/abs/1703.07402 (acedido em 21/06/2021) (ver p. 19).

[44] *World Padel Tour*. Accessed: 2020-02-21. URL: https://www.worldpadeltour.com/en/world-padel-tour/ (ver pp. 1, 6).

I

Conjunto de dados recolhido

No anexo presente está presente para cada região uma anotação correspondente:

Figura I.1: Exemplo de Anotação para Região 0

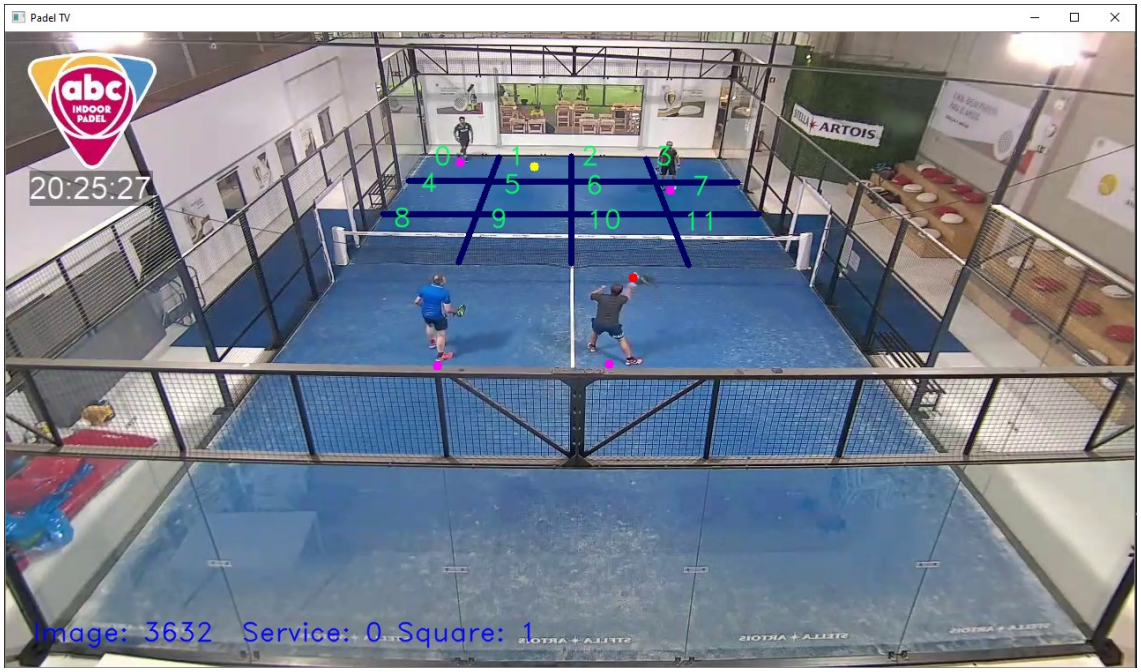


Figura I.2: Exemplo de Anotação para Região 1

Figura I.3: Exemplo de Anotação para Região 2

Figura I.4: Exemplo de Anotação para Região 3

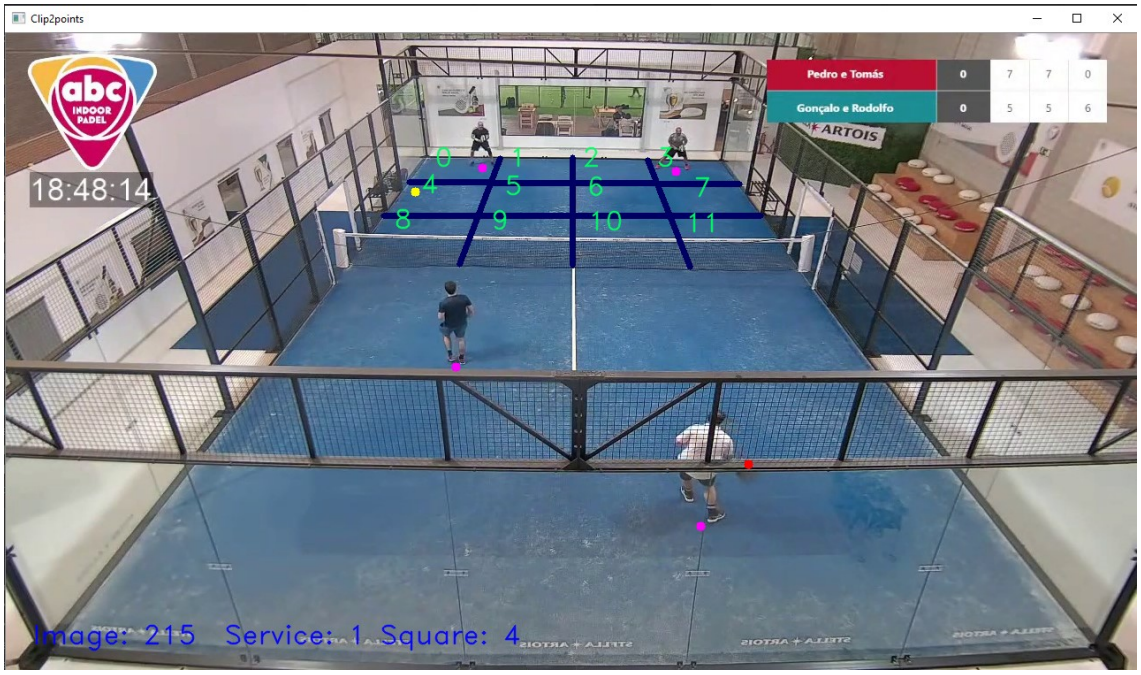


Figura I.5: Exemplo de Anotação para Região 4

Figura I.6: Exemplo de Anotação para Região 5

Figura I.7: Exemplo de Anotação para Região 6

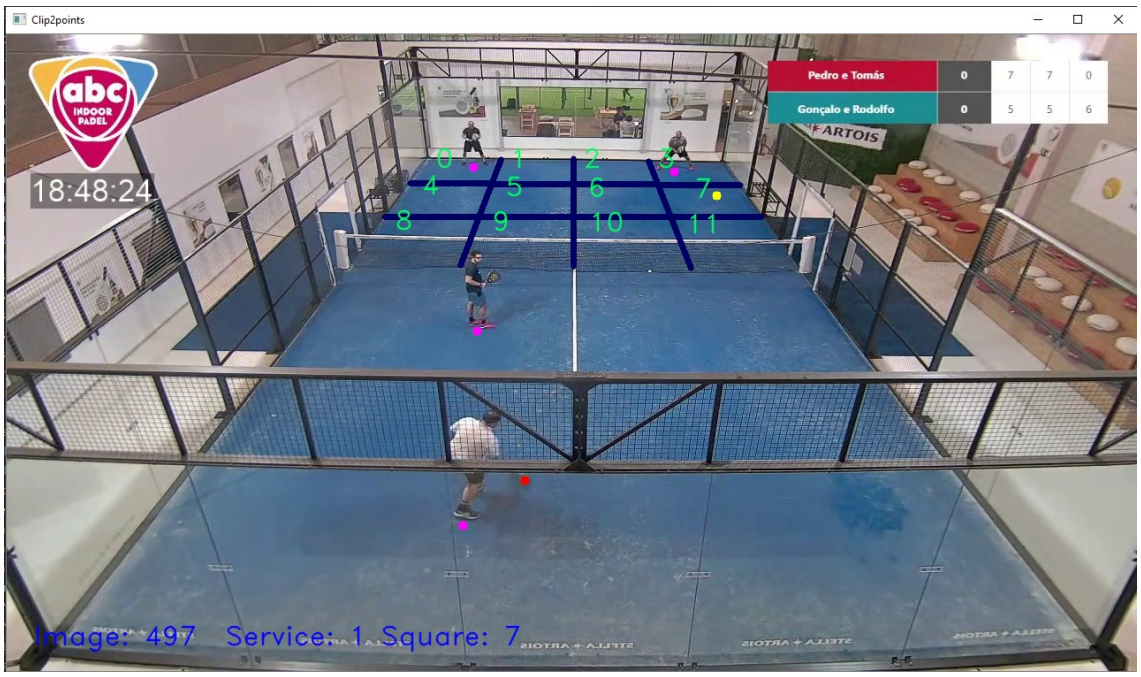


Figura I.8: Exemplo de Anotação para Região 7

Figura I.9: Exemplo de Anotação para Região 8

Figura I.10: Exemplo de Anotação para Região 9

Figura I.11: Exemplo de Anotação para Região 10

Figura I.12: Exemplo de Anotação para Região 11