Ricardo Manuel Rodrigues Amaral

Licenciado em Engenharia Informática

Optimização da Experiência de Navegação com Análise Comportamental e Aprendizagem Automática

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática

Orientador: Nuno Vasco, CTO, t_insight

Co-orientador: Ricardo Gonçalves, Professor Auxiliar, Faculdade de Ciências e Tecnologia, Universidade NOVA de Lisboa

Júri

Presidente: Prof. Doutora Teresa Isabel Lopes Romão

Vogal: Prof. Doutor Rui Pedro da Silva Nóbrega

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Fevereiro, 2021

Optimização da Experiência de Navegação com Análise Comportamental e Aprendizagem Automática

Copyright © Ricardo Manuel Rodrigues Amaral, Faculdade de Ciências e Tecnologia, Universidade NOVA de Lisboa.
A Faculdade de Ciências e Tecnologia e a Universidade NOVA de Lisboa têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objetivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "novathesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Resumo

Nos dias de hoje, existe cada vez mais a necessidade de apresentar, da forma mais direta possível, a informação que o utilizador procura. Por isso, traçar o perfil do utilizador transformou-se numa tarefa importante e determinante para qualquer plataforma ou *website* com elevado volume de informação e perante um vasto universo de utilizadores com diferentes interesses.

O problema abordado consiste na dificuldade em definir o perfil de um utilizador, sobre o qual se tem pouca informação, de forma a oferecer a melhor User Experience (UX) possível.
Este requisito é de extrema importância para a Federação Portuguesa de Basquetebol (FPB), pois com a quantidade de competições, géneros, escalões ou épocas diferentes e tendo em conta que cada utilizador, dependendo da sua localização, idade ou preferências, está interessado em diferentes tipos de resultados, estatísticas, filtros de pesquisa, entre outros, torna-se essencial proporcionar uma experiência dedicada e de maior valor.

O objetivo final deste projeto é desenvolver um sistema, a ser integrado no contexto da renovação da plataforma digital da FPB, com vista a melhorar a UX, tornando-a única e personalizada.

A solução desenvolvida para este problema passou pela extração e tratamento de informação proveniente de todas as interações do utilizador no *website*, a afetação de módulos já existentes ou criação de novos módulos para cada tipo de utilizador definindo e, por fim, recorrendo à aprendizagem automática, tornar todo o processo automático, para que o perfil do utilizador seja detetado, apenas com base nas suas ações, sem que o utilizador nos tenha de dizer o que procura no *website*.

Palavras-chave: Perfil, UX, Módulos Dinâmicos, Recolha de Dados, Aprendizagem Automática, Processamento de Dados

Abstract

These days there's an increasing need for presenting information users are looking for in the most direct way. For that purpose, building the user profile has become an important step for any company or website with a high volume of data and a large number of different types of users with different interests.

The problem we targeted consisted in building a user profile while having little information about the user in order to provide the best UX possible.

This requirement is extremely important for the FPB website because with a large amount of different competitions, genders, echelons and considering that every user has a different taste, location or age, that will culminate in different results, statistics or search parameters amoung others, it is essential to provide a dedicated and valuable experience.

The main goal of this project is to develop a system, to be integrated in the new digital platform of FPB, improving the user experience making it unique, customized and suited for a specific user profile.

Our solution to this problem went through the extraction and treatment of information from all interactions of the user on the website, the allocation of existing modules or the creation of new modules for each type of user and, finally, using machine learning, making the whole process automatic, so that the user's profile is detected, just based on their actions, without any input from the user on what they are looking for on the website.

Keywords: Profiling, UX, Dynamic Modules, Data Collecting, Machine Learning, Data Processing

Índice

Lista de Figuras

Listagens

Siglas

AJAX Asynchronous Javascript and XML

AWS Amazon Web Services

CSS Cascading Style Sheet

CSV Comma Separated Value(s)

DOM Document Object Model

FPB Federação Portuguesa de Basquetebol

HTML HyperText Markup Language

HTTP Hyper Text Transfer Protocol

IoT Internet of Things

IP Internet Protocol

JS JavaScript

MIT Massachusetts Institute of Technology

UX User Experience

Capítulo 1

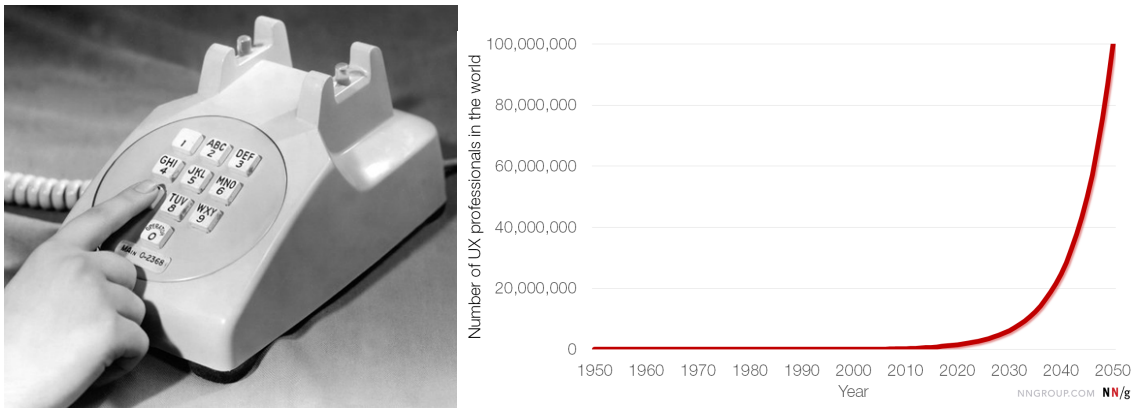
Introdução

1.1 Motivação e Contexto

Atualmente, e com a quantidade de informação com que temos contacto, a toda a hora, existe cada vez mais, da parte das empresas, a necessidade de oferecer uma experiência personalizada, dedicada e diferenciadora aos utilizadores do seu *website*. De tal modo que traçar o perfil do utilizador transformou-se numa tarefa importante para qualquer plataforma que tenha muita informação e utilizadores com diferentes interesses. Através do perfil do utilizador é possível prever o género de interação que este vai ter com o *website* e com isso desenhar o *website* de modo a melhorar a experiência do utilizador. O nível competitivo entre empresas é enorme e promover uma melhor experiência ao utilizador faz a diferença e tem, sem dúvida, um papel crucial na construção de um *website*.

A ideia de promoção de uma melhor experiência para o utilizador, apesar de mais relevante que nunca nos tempos que correm, não é nova. Foi em 1993 que Don Norman usou pela primeira vez a expressão UX, *user experience*, no seu grupo Apple Computer. No entanto, neste caso, a ideia é bastante mais antiga do que a expressão. A Nokia Bell Labs foi uma das empresas pioneiras nesta área quando contratou um psicólogo, John E. Karlin, para trabalhar no desenho do seu sistema de telefones em 1945. Em 1950 esta empresa lançou um modelo de teclado, representado na figura 1.1a, que é sem dúvida uma prova de um bom trabalho na experiência do utilizador, sendo que esses teclados são usados ainda nos dias de hoje [11]. O foco das empresas na contratação de trabalhadores em

UX tem vindo a aumentar e com tendência para continuar a subir de forma exponencial, como se pode ver na figura da direita em 1.1b.



(a) Teclado inventado por John E. Karlin

(b) Número de trabalhadores em UX de 1950-2050

Figura 1.1: Exemplo de sucesso de UX, assim como a sua evolução prevista.

A personalização é um dos fatores que promovem uma boa experiência para o utilizador. A ideia de personalização também não é nova nem exclusiva ao âmbito tecnológico. Um bom exemplo fora do mundo tecnológico é o que a distribuidora de cafés Starbucks faz no tratamento com o utilizador. Antigamente a empresa colocava nos copos o nome da bebida. No entanto, a partir de 2012 passaram a colocar o nome de cada cliente no seu copo de café. Apesar de simples, o sucesso desta mudança foi um claro exemplo de como a personalização é importante para o cliente [4]. Ian Cranna, vice presidente de *Marketing* do Reino Unido e Irlanda refere acerca desta mudança: "Os clientes da Starbucks já esperam o melhor café, mas disseram-nos que é pela ligação emocional das lojas que a Starbucks se destaca. [...] Esta campanha realça a necessidade dos clientes se sentirem únicos num mundo cada vez mais impessoal.".

No contexto de um *website*, a personalização é o processo de moldar a experiência do utilizador de acordo com os seus gostos e necessidades [14].

Existem vários exemplos desta personalização no meio digital:

- Vendedores podem direcionar ofertas aos clientes com base nos seus interesses.
- Companhias de viagens podem sugerir destinos com base em viagens feitas pelo utilizador no passado.
- Os média podem direcionar notícias com base no sítio onde o utilizador vive.

Personalização e UX, são os dois grandes objetivos que pretendemos atingir com este trabalho.

Este projeto foi inserido no contexto da renovação da plataforma digital da FPB. Esta

renovação da plataforma digital da FPB passou por um total redesenho do *website*, transformando assim um *website* algo limitado e com um *design* desatualizado, num *website* mais fresco, *responsive*, isto é, um *website* que se adapta a todos os tamanhos de ecrã (telemóvel, *tablet* ou computador), e com capacidade de responder melhor às necessidades dos seus utilizadores.

Para compreender a importância e a dimensão do projeto, é necessário ter em conta o contexto do basquetebol em Portugal. O basquetebol chegou a Portugal em 1913, pelo professor de Educação Física suíço Rodolfo Horney. A 17 de Agosto de 1927 foi fundada a FPB, na cidade do Porto. Em Portugal o basquetebol é dos desportos com mais praticantes federados, apresentando desde 2009 valores a rondar os 40.000 atletas federados, segundo as fontes IPDJ/ME, PORDATA. Naturalmente a quantidade de praticantes faz com que mais gente fique envolvida no desporto, desde adeptos a familiares de atletas, passando, naturalmente, pelos responsáveis.

O *website* da FPB é a plataforma oficial de propagação e apresentação de toda a informação relativa ao mundo do basquetebol Português. Atualmente o *website* conta com estatísticas de acessos em ordens de grandeza bastante generosas, como se pode ver na figura 1.2.

	Entradas	PageViews
Mês	**220.000**	**872.000**
Média Diária	**7.333**	**29.000**
Picos (dias de jogos)	**13.200**	**71.600**

Figura 1.2: Algumas estatísticas referentes ao número de acessos e número de páginas visitadas no *website* da FPB .

Como podemos ver na figura 1.2, o *website* da FPB conta com uma média de cerca de 7.000 pessoas diariamente sendo que esse valor chega perto do dobro em dias de jogos das principais competições. No total do mês o número de páginas visitadas chega a rondar as 900.000, o que para um *website* de interesse maioritariamente nacional é um valor bastante elevado.

1.2 Definição do Problema

Na maioria dos sistemas existentes, o utilizador tem uma conta associada e desse modo torna-se mais fácil guardar informação referente à sua navegação e posteriormente afetar que conteúdo e de que forma este lhe é apresentado. O problema que vamos abordar

consiste na dificuldade em definir o perfil de um utilizador do *website* para um utilizador sobre o qual temos pouca, ou nenhuma, informação prévia, de forma a oferecer a melhor experiência possível. Esta melhor experiência engloba todos os módulos gráficos que são necessários construir, ou alterar, para que essa experiência mais personalizada e diferenciadora seja atingida. Este problema afeta o *website* da FPB pois, com a quantidade competições, géneros, escalões ou épocas diferentes e tendo em conta que cada utilizador, dependendo da sua localização, idade ou preferências, está interessado em diferentes tipos de resultados, estatísticas, filtros de pesquisa, entre outros. Este nível de personalização é um ponto que falta ao *website* da FPB, já que o *website* tem tanta informação que é necessário conhece-lo bem para conseguir tirar o máximo potencial do *website*.

1.3 Objetivos

O objetivo final deste projeto é desenvolver um sistema, a ser integrado no contexto da renovação da plataforma digital da FPB, com vista a melhorar a experiência do utilizador, tornando-a única, personalizada e adequada ao perfil do utilizador. Como consequência, existe também o objetivo de melhorar os indicadores de performance do *website* da FPB, nomeadamente o tempo de sessão, número de acessos, e o número de interações e partilhas de conteúdo. Em relação à secção da loja do *website* da FPB, existe também o objetivo de potenciar vendas e valorizar espaço publicitário.

Em termos práticos, são estes os objetivos que se pretende atingir ao longo deste projeto:

- Implementação de mecanismos de extração e filtragem de dados relevantes sobre o utilizador.
- Avaliação e construção de utilizadores tipo, através da deteção de comportamentos refletidos nos dados recolhidos.
- Refletir o tipo de utilizador em blocos dinâmicos e funcionais que respondam graficamente às decisões tomadas pelo modelo anterior. Isto inclui criação de novos módulos, afetação de menus de navegação, criação de atalhos que façam o utilizador saltar páginas sem interesse para o mesmo.
- Desenvolvimento de um programa de aprendizagem automática que permita, com base em informação recolhida, encaixar o utilizador num utilizador tipo de forma dinâmica, isto é, variável à medida que o utilizador navega, e automatizada.

1.4 Solução Proposta

Para abordar o problema apresentado, este foi dividido em quatro partes: recolha de dados, definição de perfis, desenvolvimento de módulos dinâmicos e desenvolvimento de

algoritmias. Em relação à recolha e enriquecimento de dados, será necessário recolher o máximo de informação possível sem que seja intrusivo para o utilizador. O segundo ponto passa pela definição de perfis e de comportamento padrão. Isto é, cada tipo de utilizador tem um comportamento diferente quando navega no *website* da FPB, ou seja, diferentes áreas de interesse no *website* e diferente tipo de conteúdo. Por exemplo, um utilizador pode estar mais interessado nas notícias da associação de basquetebol da sua região, enquanto outro dá preferências aos últimos documentos ou comunicados feitos pela federação. É necessário, por isso, conhecer os utilizadores e analisar, com base nos dados recolhidos, os seus comportamentos padrão. No terceiro ponto, após a extração de dados e consequente definição do utilizador, são criados módulos dinâmicos que refletem a informação recolhida em alterações visíveis e com relevância para os utilizadores. Estas alterações vão desde criação de atalhos, que facilitem a navegação ao utilizador, à criação de módulos com vários tipos de conteúdo de forma mais concentrada, tentando juntar grande parte do conteúdo que o utilizador procura, num só *"Widget"*.

No último ponto, com vista a minimizar a necessidade de ser o utilizador de forma manual a informar-nos que tipo de utilizador é ele, foi também criado um programa de aprendizagem automática que fará a associação entre os perfis criados e os dados recolhidos. Desta forma, o tipo de utilizador é um parâmetro variável ao longo da navegação no *website*, sendo que ao longo da sessão, os dados extraídos, continuam a ser enviados ao algoritmo de aprendizagem automática, de forma a detetar se o perfil do utilizador se alterou.

Resumindo, para captar a atenção do utilizador e tornar a sua experiência o mais completa e personalizada possível, é necessário então uma forma de saber o que cada utilizador procura à medida que este interage com o *website*. Por isso, a solução para este problema passa pela extração e tratamento de informação proveniente de todas as interações, comportamentos, proveniências, localização, entre outros. O resultado da identificação do perfil ou interesses do utilizador resulta na afetação de conteúdo que permita melhorar a experiência do utilizador do *website* da FPB. De modo a automatizar todo o sistema, sem necessitar que o utilizador nos diga que tipo de utilizador ele é, criámos o algoritmo de aprendizagem automática, para que este encaixe os dados recolhidos num perfil de utilizador.

Capítulo 2

Trabalho Relacionado

Esta secção apresenta o trabalho relacionado, assim como, de forma mais detalhada, as ferramentas e tecnologias que são usadas na implementação do projeto. Serão ainda referidos conceitos essenciais para uma melhor compreensão do mesmo. O projeto foi dividido em quatro fases, desse modo, vamos abordar o trabalho relacionado nessas quatro vertentes, **Recolha e Enriquecimento de Dados**(2.1), **Definição de Perfis e Comportamentos Padrão**(2.2),**Produção de Interface** (2.3) e **Desenvolvimento de Algoritmias e Autonomias** (2.4).

Para demonstrar o que se pertende quando dizemos que queremos melhorar a experiência do utilizador no *website* da FPB, vamos começar por mostrar exemplos de empresas que se destacam pelos ótimos resultados que as suas plataformas têm, muito devido aos sistemas de optimização da navegação dos utilizadores nos seus *websites*.

Existem várias empresas na vanguarda dos sistemas de sugestão. Algumas usam técnicas que são usadas nesta tese, outras devido a alguns fatores, que vão ser referidos de seguida, conseguem levar a UX a um outro nível.

Um exemplo de uma empresa, na linha da frente neste tipo de tecnologia, é a Google. Mais concretamente o caso do Youtube. O Youtube utiliza um algoritmo de aprendizagem automática com excelentes resultados. É um algoritmo que tem vindo a ser melhorado, a cada ano que passa, e isso reflete-se na quantidade de utilizadores que utilizam a plataforma. Mas como é que o Youtube chegou aqui? Antes de 2012, o Youtube usava apenas uma métrica para classificar os vídeos, as visualizações. Esta tática tinha como objetivo beneficiar, supostamente, os melhores vídeos, no entanto, acabou por trazer um problema de *clickbait*, isto é, os criadores exageravam os títulos dos vídeos e levavam os utilizadores

a abrir o vídeo, para só aí perceberem que não era o que esperavam.

De 2012 a 2016, o Youtube introduziu duas novas variáveis para colmatar o problema do *clickbait*, o tempo de visualização e o tempo de sessão. Apesar das melhorias, esta solução trouxe também novos problemas. Criadores usavam uma estratégia de retardamento do conteúdo do vídeo. Com introduções muito grandes, para que o utilizador ficasse mais tempo no vídeo. Outro problema que esta solução trouxe foi beneficiar os vídeos longos, o que é um problema para os criadores, pois têm de tentar manter a qualidade.

O algoritmo do Youtube tem vindo, até aos dias de hoje, a ser melhorado e optimizado. Apesar de algumas dificuldades, nomeadamente em bloquear conteúdo sensível, o Youtube é um grande exemplo no que ao seu sistema de sugestão de vídeos e à sua UX, com resultados evidenciados em números, cerca de 1 bilião de horas de visualização de vídeos, por dia.

Um fator que simplifica o trabalho do Youtube, quando comparado com o que se vai desenvolver nesta tese, é o facto de só haver um tipo de conteúdo, vídeos, e esse conteúdo ser totalmente categorizável. Apesar de secreto, o algoritmo do Youtube funciona tanto com sessão iniciada, onde sugere vídeos baseados na história do utilizador, como sem sessão iniciada, onde para além de mostrar os vídeos em destaque, consoante a localização do utilizador, mostra ainda sugestões com base nos vídeos que o utilizador vai vendo ao longo da sessão [6]. A solução, no caso do Youtube, foi utilizar aprendizagem automática para fazer uma mistura entre vídeos já navegados, tendo em conta o tempo de visualização e outros fatores, e os vídeos em destaque, tendo em conta a localização do utilizador, e daí prever que vídeos o utilizador estará interessado. Também no caso da FPB, estamos perante algo semelhante, na medida em que procuramos uma solução que permita, com base nas ações do utilizador e da sua proveniência, detetar qual o seu perfil. Utilizando aprendizagem automática, é possivél ultrapassar esta dificuldade sem que seja necessário questionar o utilizador em relação aos seus gostos.

Outro exemplo de uma empresa que usa aprendizagem automática para optimizar o seu *website* é a Amazon. A Amazon faz uma análise tendo em conta artigos que o utilizador comprou, já os tem ou avaliou. A atividade do utilizador na página é comparada com outros utilizadores. O algoritmo da Amazon dá relevância ao facto de que se dois utilizadores têm interesse em comum num artigo, então os seus interesses podem-se sobrepor noutros artigos. Este é um processo continuo de avaliação dos interesses dos utilizadores, interesses esses, que mudam ao longo do tempo [3]. Esta interligação de interesses vai ser abordada também nesta tese, pois no caso da FPB, utilizadores que, por exemplo, acederam à mesmas notícias, podem ter os mesmos interesses. Outro ponto onde o trabalho da Amazon e o que vai ser desenvolvido nesta tese converge, é na afetação de módulos dinâmicos. A página inicial da Amazon não é igual para todos os utilizadores, existem

secções, que são diferentes para utilizadores diferentes. Deste modo, o *website* é praticamente construído à medida de cada utilizador.

Um conceito importante de salientar antes de avançarmos para a recolha enriquecimento de dados, é a estrutura do *website* da FPB. O *website* da FPB foi construído por cima da plataforma *WordPress*.

O *WordPress* é uma plataforma muito utilizado na construção de *websites*, cerca de 39% dos *websites* são construídos usando o *WordPress*, desde pequenos *bloggers* a grandes marcas mundiais. O *WordPress* é usado, principalmente em *websites* dinâmicos, isto é, *websites* em que diariamente é necessário adicionar ou alterar conteúdo. Isto acontece devido ao facto da ferramente *WordPress* estar dividida em duas partes, a parte visual do *website*, 2.1, e um *Backoffice*, 2.2, uma ferramenta do *webiste* ao qual apenas utilizadores autorizados têm acesso e onde é possível alterar e criar conteúdo de forma rápida e fácil. Outra grande vantagem do *Wordpress* é o facto de permitir, de forma bastante flexível, a criação de tipos de conteúdo, denominados *"post_type"*, assim como de categorias ou *"terms"*, dentro desse tipo de conteúdo. Este ponto foi extremamente importante na categorização de todo o *website* pois apenas podemos fazer um trabalho tão profundo na otpimização e apresentação dos conteúdos que o utilizador procura porque esses conteúdos estão todos categorizados.

Figura 2.1: *Website* da FPB

,

2.1 Recolha e Enriquecimento de Dados

Numa primeira fase, é imperativo uma recolha adequada de informação. Definir que informação é relevante, escolher os métodos de recolha mais eficientes para o caso concreto da FPB, processar esses dados, filtrar exceções, anomalias, assim como comprimir os dados

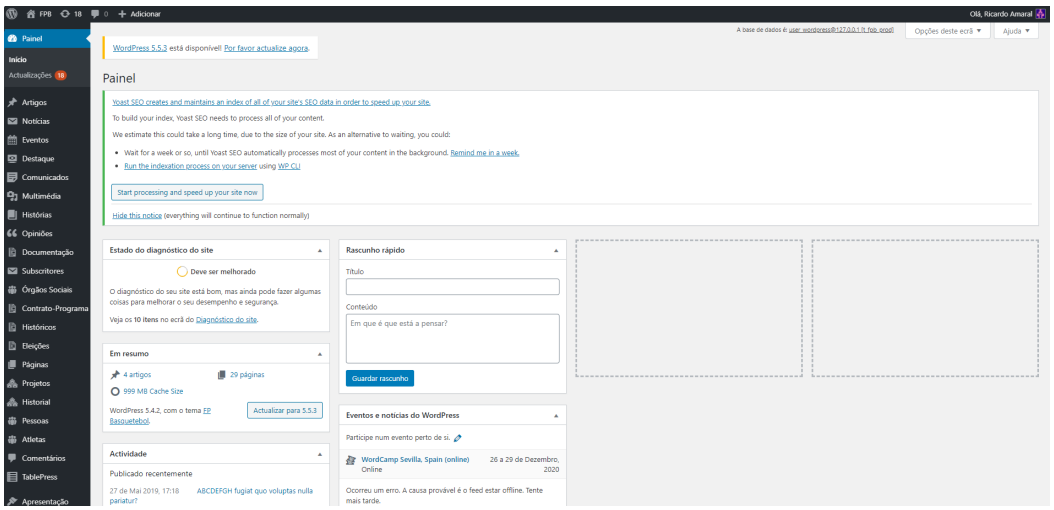


Figura 2.2: *Backoffice* do *website* da FPB

finais para que estes fiquem prontos para serem servidos aos algoritmos de aprendizagem automática. Este é um dos passos mais desafiantes do projeto.
Como se pode então obter a informação que se pretende?

2.1.1 Registo de Atividade

Quando um cliente faz um pedido de uma página ao servidor, existe um fluxo de dados do cliente para o servidor e vice-versa. Esses dados não são visíveis ao utilizador comum, pois são encapsulados no pedido Hyper Text Transfer Protocol (HTTP). Num pedido HTTP o cliente constrói um pacote com a informação necessária para que o servidor, ao receber, esse pacote o consiga abrir e identificar qual o pedido do cliente e agir em conformidade. Nesse pacote existem três principais secções, Pedido, Cabeçalho e Corpo [12]. Para a recolha de informação a secção importante é o Cabeçalho, sendo que esta pode conter informação relevante sobre o utilizador.

Informação presente no cabeçalho de um pedido http:

- O **endereço de Internet Protocol (IP)** do utilizador. Este é um identificador do utilizador na rede. Apesar de não ser um identificador estático e único por utilizador, pode ser usado como fonte de informação.

- **Endereço Referente**, que é usada para obter informação sobre de que página o utilizador veio. Esta informação, com um número grande de utilizadores e ao longo do tempo permite a criação de mapas que podem ajudar a prever e sugerir para que página o utilizador tende a navegar de seguida.

- **Marca Temporal**. É uma valor universal que nos permite saber o momento exato em que o utilizador formulou o pedido HTTP. Manter esta informação pode ser

bastante relevante para tirar conclusões sobre quanto tempo um utilizador passa em cada página o que pode ser um indicador do perfil do utilizador.

- **Autenticação**. Quando disponível esta informação permite saber, inequivocamente, que utilizador está a fazer o pedido. Com o utilizador identificado o perfil do mesmo é imediatamente obtido, caso as preferências do utilizador tenham sido guardadas em sessões anteriores.

- **Detalhes do navegador**. Apesar de menos relevante existe também um campo destinado a fornecer informação sobre o navegador através do qual o utilizador fez o pedido.

2.1.2 Localização

Apesar de extremamente útil, este dado é mais difícil de obter pois, por norma, para obter esta informação, é necessário o consentimento por parte do utilizador. Isto é um problema pois, cada vez mais, os utilizadores preferem não fornecer a sua localização, tirando nos casos em que isso lhes traga um benefício claro [8].

Para a FPB a obtenção da localização do utilizador é bastante relevante e útil para o utilizador, pois esta localização pode influenciar que competições o utilizador acompanha, notícias perto da sua localização, informação relevante sobre a associação de basquetebol da sua zona. Para obter esta informação será utilizada uma tecnologia da Google chamada Geocoding API [7]. Esta biblioteca permite, mediante a autorização do utilizador, obter informações como o distrito, concelho ou cidade.

2.1.3 Formulários

Existe ainda a opção de requisitar o preenchimento de formulários por parte dos utilizadores. Estes formulários são bastante importantes pois ajudam a classificar o utilizado. Estes formulários são a forma mais direta de obter a informação necessária, visto que são construídos precisamente com esse propósito. Com esta solução, praticamente, não é necessário tratamento dos dados recolhidos, visto que podem ser construídos já tendo em conta o formato necessário.

2.1.4 Sessão

Uma sessão é um mecanismo que permite ao servidor guardar informação que persiste ao longo de toda a interacção do cliente com o servidor, desde que o cliente faz a ligação ao servidor até fechar o navegador. Esta solução é uma abordagem que permite baixar o peso da quantidade de informação guardado em Cookies2.1.5, isto é, do lado do cliente. Com esta solução o peso recai do lado do servidor, sendo que, do lado do cliente, é necessário guardar apenas um identificador de sessão que permita ao servidor fazer a ligação entre

o cliente e a sua sessão [13].

No âmbito da FPB este método de identificação de sessão é extremamente útil para identificar o perfil comportamental dos utilizadores, permitindo detetar mudanças de páginas, com as respetivas marcas temporais, para assim retirar informação sobre a ordem pela qual as páginas são visitadas, assim como, a quantidade de tempo que o utilizador fica em cada página.

2.1.5 Cookies

Ao contrário da sessão onde os dados são guardados do lado do servidor as Cookies são mantidas do lado do cliente, por esse motivo, são por norma menos pesadas, isto é, guardam menos informação para não sobrecarregar o utilizador. Podem ser usadas para guardar informação referente ao identificador de sessão ou preferências gerais do utilizador. Pequenas porções de dados, que precisem de ser mantidas ao longo de várias páginas, podem também ser mantidos em Cookies, um bom exemplo disso são os carrinhos de compras *online*.

Neste projeto são usadas Cookies para manter algumas preferências como, filtros selecionados ou pesquisas feitas pelo utilizador assim como a sua localização quando obtida através do método referido anteriormente 2.1.2.

2.1.6 Ações do Utilizador

No processo de recolha de dados as ações do utilizador têm um papel preponderante, pois estas ações estão intrinsecamente ligadas às preferências, gostos ou desejos do utilizador.

Esta captura de ações ou eventos, vai ser feita com JavaScript do lado do cliente. Neste caso, usar-se-à a biblioteca jQuery, que simplifica alguma sintaxe, quando comparada com JavaScript puro. JQuery é uma ferramenta *open source* produzida pelo Massachusetts Institute of Technology (MIT), sendo neste momento a biblioteca mais usada no desenvolvimento de *websites*.

Listagem 2.1: Registo de Atividade

```
1 function eventTracker(event) {
2     if(e.originalEvent === undefined){
3         /*Evento nao provocado pelo Utilizador*/
4     }else{
5         /*Guardar informaçao relevante do evento*/
6     }
7 }
8 $(document).ready(function(){
9     $(document).bind("click␣keydown", function(event){
```

```
10          eventTracker(event);
11      });
12  });
```

Em cima, 2.1, está representado um esboço de como este mecanismo de deteção de eventos irá funcionar. O código apresentado, tem como função, detetar todos os cliques ou teclas pressionadas pelo utilizador e retornar o evento associado a essa ação. Existem vários tipos de eventos que podem ser detetados, sendo o *click*, que deteta os cliques na página, e o *keydown*, que deteta quando uma tecla é carregada, os mais importantes. No entanto, outros outros eventos podem vir a ser adicionados consoante a necessidade. Tipos de eventos que podem vir a ser adicionados:

- **dblclick** - Deteta um duplo clique na página.
- **mouseenter** - Permite detetar quando o rato fica por cima de um elemento do Document Object Model (DOM) (DOM é uma interface que permite, aos *Scripts*, aceder aos elementos de uma página).
- **mouseleave** - Permite detetar quando o rato deixa o elemento do DOM .
- **hover** - Recebe uma ou duas funções, se receber uma comporta-se como o mouseenter, se receber duas, a primeira será o mouseenter e a segunda o mouseleave.

Informação mais relevante contida num Evento [9]:

- **Target** - Retorna o elemento do DOM envolvido no evento, se existir.
- **RelatedTarget** - Se houver outro elemento do DOM envolvido no evento retorna-o.
- **PageX e PageY** - Corresponde respetivamente as coordenadas X e Y do clique do utilizador no referencial da página.
- **Which** - Para teclas indica que tecla foi pressionado, sendo que cada tecla esta associada a um valor. Para um clique com o rato indica também que botão do rato foi selecionado.
- **OriginalEvent** - Esta campo permite saber se o evento foi provocado por um utilizador ou por um Script. Isto é bastante útil para filtrar acessos não humanos.

Listagem 2.2: Retorno de um clique

```
1
2  altKey: (...)
3  bubbles: (...)
4  cancelable: (...)
5  changedTouches: (...)
6  ctrlKey: (...)
```

```
detail: (...)
eventPhase: (...)
metaKey: (...)
pageX: (...)
pageY: (...)
shiftKey: (...)
view: (...)
char: (...)
code: (...)
charCode: (...)
key: (...)
keyCode: (...)
button: (...)
buttons: (...)
clientX: (...)
clientY: (...)
offsetX: (...)
offsetY: (...)
pointerId: (...)
pointerType: (...)
screenX: (...)
screenY: (...)
targetTouches: (...)
toElement: (...)
touches: (...)
which: (...)
originalEvent: MouseEvent {isTrusted: true, screenX: 631, screenY: 380, clientX: 631,
    ↪ clientY: 310,...
type: "click"
isDefaultPrevented:
target: body
currentTarget: document
relatedTarget: null
timeStamp: 4319.680000015069
jQuery34106204356110360396: true
delegateTarget: document
handleObj: {type: "click", origType: "click", data: null, guid: 1, handler:
data: null
```

O código acima, 2.2, mostra, com um exemplo real, toda a informação que um clique na página nos pode dar.

No entanto, recolher todas as interações do utilizador com a página acabaria por selecionar demasiada informação desnecessária, que consumiria uma quantidade grande de memória. Por isso, para o processamento de dados, são detetadas apenas as interações em elementos cruciais do *website*. São relevantes, por exemplo, cliques em filtros ou em hiperligações.

2.1.7 Processamento dos Dados

Depois de recolhida, a informação necessita de ser processada e preparada para ser guardada de forma eficiente e limpa. Este processo vai decorrer num serviço à parte hospedado na Amazon Web Services (AWS), tema mais aprofundado no capítulo 2.4.6.

Para alcançar este efeito é preciso recorrer aos seguintes passos:

- **Limpar os Dados** - Nem todos os dados retirados são úteis, por exemplo, existem vários acessos a ficheiros que não tem nenhuma relevância para detetar o perfil do utilizador (ficheiros JavaScript (JS), Cascading Style Sheet (CSS), imagens, entre outros), acessos não humanos, erros. Por isso existe a necessidade de limpar os dados brutos retirados e manter apenas o essencial.

- **Identificação do Utilizador** - O melhor método de identificação é com uma conta por cada utilizador. No entanto, como referido anteriormente, essa hipótese não é, por norma, usada em *websites*, a não ser que a identificação inequívoca do utilizador seja completamente imprescindível. Por isso é necessário encontrar alternativas. Outro método também já referido anteriormente é o endereço de IP, que apesar de nos poder dar algumas informações, é um método de identificação fraco devido ao facto de vários utilizadores poderem ter o mesmo endereço de IP e ao mesmo tempo o mesmo utilizador pode mudar de endereço de IP, mesmo sem mudar de sessão. Por isso a forma mais segura de identificar o utilizador é através do seu identificador de sessão (*SessionID*) .

2.1.8 Resumo

Concluindo, e relacionando o que foi mencionado em cima com o projeto a implementado, a ideia é recolher todos os dados relevantes durante a navegação do utilizador no *website* da FPB, para que desse modo seja possível criar módulos dinâmicos.

Na figura 2.3 está representado um esboço do que será a estrutura do projeto, assim como o fluxo de dados, entre os vários componentes.

O fluxo à direita do utilizador na imagem acima será accionado de 30 em 30 segundos ou quando o utilizador muda de página para que nenhum dado seja perdido. O pedido será um pedido Asynchronous Javascript and XML (AJAX), tecnologia que permite fazer pedidos de forma assíncrona e em segundo plano. A informação será guardada em sessão, sessão esta que vai sendo atualizada pelo perfil retornado pela ferramenta de aprendizagem automática. Dentro do serviço hospedado na AWS, o processamento de dados está dividido do algoritmo de aprendizagem automática para que este receba a informação já tratada.

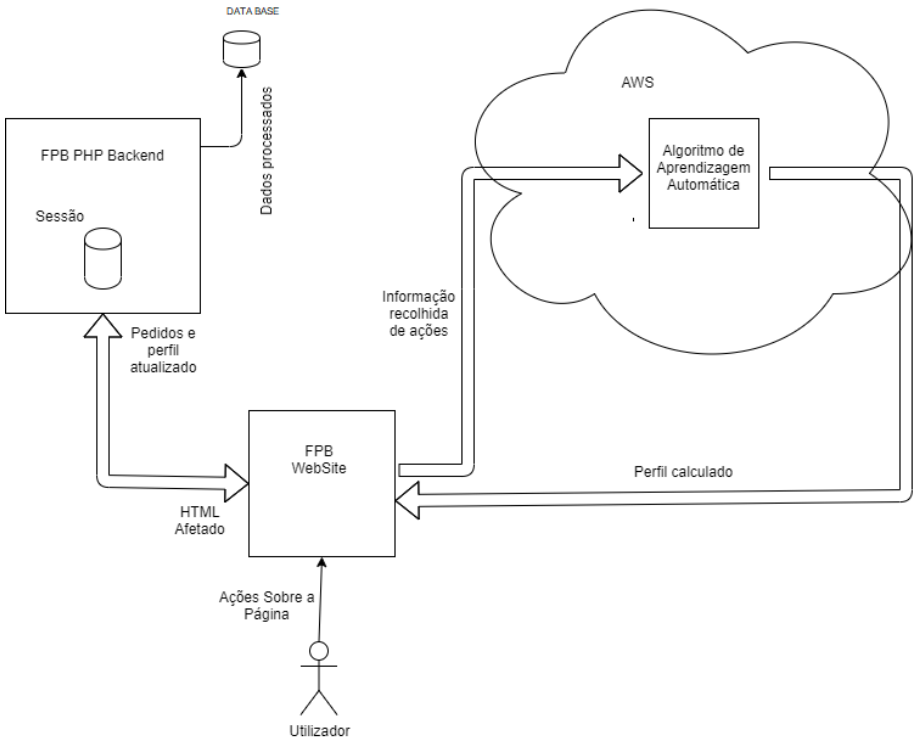


Figura 2.3: Rascunho do fluxo de dados.

2.2 Perfis e Comportamentos Padrão

Com o passar dos anos a expectativa dos utilizadores em relação ao comportamento de um *website* mudou de forma drástica. Com a evolução computacional os utilizadores são cada vez mais exigentes na relevância da informação que lhes é fornecida. O processo de criação de um perfil é um fator de grande importância no impacto que as empresas pretendem ter na personalização dos seus *websites*. Esta personalização passa por apresentar a cada tipo de utilizador uma navegação mais eficiente. Esta noção de eficiência pode ser medida pelo nível de facilidade de um utilizador em encontrar a informação que procura. Quanto mais fácil e direta for esta procura, melhor será a experiência do utilizador.

2.2.1 Conhecer os Utilizadores

O primeiro passo na construção do perfil passa por responder as seguintes questões:

- **Quem?** Que utilizadores vão aceder à plataforma. No caso da FPB prevê-se que a maioria dos utilizadores serão interessados em Basquetebol, por exemplo, jogadores, treinadores, árbitros ou adeptos.

- **Quando?** Em que altura estes utilizadores acederam. Saber se são utilizadores assíduos ou se estão a aceder por um outro motivo, por exemplo, se está a aceder perto

da hora de um jogo, também pode indicar que informação este utilizador vai desejar obter.

- **Onde?** De que localização o utilizador está a aceder, por exemplo, se estiver a aceder de Bragança pode indicar que vai estar mais interessado em artigos ou notícias sobre associação de basquetebol de Bragança.

2.2.2 Traçar Perfis

Os utilizadores têm, por norma, comportamentos distintos entre si, pelo que detetar esses comportamentos, establecer padrões e refleti-los em diferentes perfis faz todo o sentido.

O perfil do utilizador reflete o seu comportamento, objetivos e expetativas. Seguem-se cinco exemplos de perfis pré-definidos, sendo possível que, no futuro, venham a ser criados mais:

- **Jogador** - Caracteriza-se por um interesse na zona de estatísticas de uma competição específica, ou acessos ao detalhe de jogadores da sua equipa ou equipas que vai defrontar num futuro próximo. Este perfil de utilizador tem um maior foco em estatísticas, logo a criação de módulos e resumos estatísticos é importante.

- **Treinador** - Caracteriza-se por um interesse no historial de uma equipa, estatística sobre a sua equipa ou equipas que vai defrontar. Para este perfil de utilizador poder-se-à criar módulos com o historial de jogos de uma equipa e com as respetivas estatísticas assim como disponibilizar notícias relativas à sua equipa ou equipas que vai jogar no futuro assim como informação relativa a treinadores.

- **Árbitro** - Acessos a documentação, novas directrizes, notícias relativas a árbitros, são as características que definem este tipo de utilizador. Faz sentido criar um módulo que permita o acesso rápido às documentações relativas a árbitros, nomeações e castigos, por exemplo.

- **Tendências** - Caracteriza-se por um utilizador que segue as notícias, competições, atletas mais mediáticos. Para este tipo de utilizador faz sentido apresentar com maior relevância as páginas ou elementos mais acedidos pela maioria dos utilizadores.

- **Novo** - Um utilizador novo caracteriza-se por uma procura e investigação do *website* num todo, mais tempo em cada página, acesso a várias páginas com termos diferentes. A este utilizador faz sentido apresentar a maior variedade diferente de informação para que, com acesso à informação toda, este consiga definir os seus interesses, caso volte a entra no *webiste* da FPB.

Existem principalmente duas formas de obter o perfil do utilizador:

- **Filtragem com base no conteúdo** - Este tipo de filtragem, também conhecido como filtragem cognitiva, passa por comparar os termos do objeto que o utilizador selecionou com os termos associados a um perfil de utilizador. Ou seja, tendo como exemplo, um perfil designado "Informático" com os termos "Aprendizagem Automática", "*Internet of Things (IoT)*" e "*Cloud*". Quando um utilizador navega por várias páginas com esses termos, poder-se-à começar a associar o utilizador ao perfil "Informático", prevendo assim que os interesses do utilizador estarão associados à área com mais termos em comum. Este foi um exemplo de filtragem implícita, existe ainda a filtragem explicita, onde através de um mecanismo de voto, por exemplo, um utilizador poderá classificar um produto. Este segundo método baseia-se na comparação de termos diretamente na votação do utilizador enquanto que o método implícito tem por base a navegação [10].

- **Filtragem colaborativa** - . Esta passa por criar grupos de utilizadores e assumir que os utilizadores que seguiram os mesmo termos no passado vão continuar a seguir no futuro. Quanto mais forte for a diferença entre grupos maior a probabilidade de sucesso. Técnicas alternativas envolvem usar dados geográficos ou demográficos para criar os grupos, assumindo assim que pessoas da mesma área, faixa etária ou do mesmo género, por exemplo, vão ter gostos semelhantes [10].

- **Filtragem Direta** - . Esta filtragem passa por utilizar, de forma direta, *inputs* do utilizador. Por exemplo, termos que o utilizador insere na ferramenta de pesquisa do *website* podem ser associados, quase de forma direta, aos termos chave de interesses daquele utilizador.

2.3 Produção de Interface

Esta será a última fase do projeto e consiste na afetação ou criação de módulos no *website* da FPB. É neste passo que o utilizador vai, de facto, tirar benefício de todo o processo anterior. É portanto o passo onde se vai detetar o quanto esta ferramente vai ajudar o utilizador. É por isso muito importante definir:

- Que tipo de conteúdo pode ser alterado.

- Qual a melhor forma de alterar esse conteúdo.

- Que módulos podem ser criados para ajudar o processo de optimização da navegação do utilizador.

- Onde é que os módulos criados vão aparecer ao utilizador.

No entanto antes de análisar o caso concreto da FPB, é necessário perceber, o que se entende por conteúdo dinâmico.

2.3.1 Conteúdo Dinâmico

O processo de criação de conteúdo dinâmico passa pela avaliação comportamental do utilizador. No âmbito de um *website* é expectável que blocos de informação sejam afetados pelos interesses e preferências do utilizador.

Existem duas formas principais de formar este conteúdo:

- Baseado em informação recolhida em sessões anteriores.
- Afetando esses blocos com base em informação recolhida durante a sessão atual.

Para que o conteúdo gerado seja totalmente preciso, no primeiro caso, por norma, é necessário existir uma noção de sessão por parte do utilizador para que este seja identificado inequivocamente. Deste modo a informação dentro dos blocos dinâmicos será totalmente direccionada às preferências do utilizador. Isto permite que na construção do HyperText Markup Language (HTML) da página existam secções de conteúdo dinâmico, automaticamente preenchidas com o perfil de cada utilizador.

Existem vários exemplos deste tipo de conteúdo em praticamente todos os *websites* com sessão. Dois exemplos fortes desta prática são a Netflix e o Instagram. Estas companhias dão a maior importância ao perfil e preferências do utilizador. Isto faz com que os algoritmos de sugestão de conteúdo sejam extremamente coerentes, acabando por oferecer aos utilizadores uma melhor experiência.

Estas empresas levaram a noção de conteúdo dinâmico para outro patamar, onde deixa de ser só a informação pessoal de cada utilizador a ser construída de forma dinâmica para uma ideia de campos de sugestão dinâmica, isto é, enormes blocos de notícias, pessoas, locais, entre outros, que são todos eles dinâmicos e baseados nos interesses do utilizador. Este modelo levanta no entanto algumas questões de privacidade, pois o facto da identificação do utilizador ser única faz também com que essa informação seja pessoal, o que nos deixa numa posição complicada onde é preciso encontrar um equilíbrio entre dados pessoais e dados relevantes para a personalização de conteúdo para o cliente.

Para além de informação sobre um utilizador, conhecido do sistema, existe ainda a construção de conteúdo dinâmico em tempo real, as ações do utilizador, afetam de forma direta, o comportamento da página. Nestes casos existe uma dificuldade acrescida pois o sistema, inicialmente, não sabe as preferências do utilizador, por isso é necessário detetar todas as interações do utilizador com a página para assim tentar prever o seu perfil.

Entre exemplos de afetação de módulos temos:

- **Pré preenchimento** - Possibilidade de afetar filtros consoante os interesses do utilizador, por exemplo, se foi detetado que um utilizador tem interesse na liga feminina, quando acede a uma página, onde seja possível filtrar por essa liga, os filtros estarem automaticamente selecionados.
- **Menu** - Alterar menus de navegação de forma a apresentar primeiro os acessos as páginas que o utilizador pode ter mais interesse, agrupar grupos ou remover opções.
- **Ordem** - Alterar ou remover módulos da página consoante o perfil traçado do utilizador.
- **Loja** - Apresentação, em seções destinadas a contacto promocional, de artigos, presentes na loja da FPB. Estes artigos são direcionados, consoante a informação recolhida sobre o utilizador.

2.3.2 Criação de Módulos

Este projeto vai passar ainda pela criação de novos módulos consoante o perfil do utilizador previamente traçado. Esses podem ir da criação de novas ligações para servir como atalhos para o utilizador a resumos com agregações de informações, como estatísticas para que o utilizador obtenha a informação que procura com o mínimo de navegação possível.

Este novos módulos, com resumos de informação, relevante ao utilizador, passam, por exemplo, no caso de um utilizador com interesse na Liga Feminina, pela apresentação de algumas estatísticas globais da competição, destacar jogos com maior interesse, assim como, as jogadores com melhor nível de performance nos últimos jogos. Esta agregação traz vantagens ao utilizador, pois conseguirá consumir informação que, num caso normal, estaria dispersada por várias páginas ao longo do *website*, num pequeno bloco, condensado de informação, acessível em qualquer página do *website*.

2.4 Algoritmias e Autonomias

Após a recolha de dados e a definição de perfis procedeu-se então ao desenvolvimento de uma ferramenta que permite a partir dos dados recolhidos chegar aos perfis definidos. Neste capítulo será abordado o tema da aprendizagem automática assim como as principais ferramentas e os seus prós e contras.

2.4.1 Aprendizagem Automática

A aprendizagem automática é uma ferramenta de grande importância no mundo empresarial dos nossos tempos, pois permite treinar os sistemas computacionais de uma forma automática e com pouca, ou nenhuma, supervisão humana. Os computadores de hoje são

cada vez mais rápidos. Isto é verdade não para todas as tarefas, mas em tarefas como a deteção de padrões as máquinas são muito superiores aos humanos, principalmente pela velocidade com que tratam quantidades muito elevadas de dados. Outra vantagem é a sua capacidade de ignorar informação irrelevante.

Baseado num modelo de comunicação de neurónios criado em 1949 por Donald Hebb num livro *The Organization of Behavior*, a aprendizagem automática usa algoritmos e redes neuronais para ajudar os sistemas de computadores a melhorar a sua performance, de forma progressiva. Traduzindo o conceito de Donald Hebb para redes neuronais artificiais, o seu modelo pode ser descrito como uma forma de afetar as ligações neuronais, sendo que essa ligação fica fortalecida quando os dois neurónios são chamados ao mesmo tempo e enfraquece quando são chamados separadamente.

Deste 1950 que a aprendizagem automática tem crescido de forma exponencial, foi neste ano que Arthur Samuel, um programador da IBM desenvolveu um programa para um jogo de xadrez. O seu programa baseava-se numa função mínimo máximo, que tentava calcular de que forma a próxima jogada influenciaria cada lado do tabuleiro escolhendo depois a que minimizava a pontuação do adversário e aumentava a sua. No entanto só em 1952 é que Arthur Samuel cunha a expressão *Machine Learning* em português, aprendizagem automática.
Seguiram-se ao longo dos anos vários usos para esta ferramenta, destacando-se o reconhecimento de voz e facial. Hoje em dia a aprendizagem automática afeta principalmente estes campos:

- Análise de dados de vendas
- Programas de decisão
- Processamento de linguagem natural
- Deteção de fraudes
- Preços dinâmicos, com base na oferta/procura
- Personalização em tempo real
- Recomendação de produtos

Estes dois últimos pontos são os mais importantes no desenvolvimento da ferramenta de aprendizagem automática para o *website* da FPB. Aprendizagem automática é útil em projetos onde se pretende prever um *output* ou detetar padrões. O uso desta ferramenta ganha força, quando comparada com simples regras de associação entre as interações do utilizador e o seu perfil. Nomeadamente porque é espectável que o *website* da FPB venha a ter um número elevado de acessos, gerando, consequentemente, um número elevado de

combinações possíveis, pelo que seria impraticável definir todas essas regras, com pesos diferentes e múltiplas combinações.

2.4.2 Como Funciona?

Os sistemas de aprendizagem automática são constituídos por três partes principais:

- **Modelo** - A parte do sistema responsável por prever ou identificar o resultados. Esta é a primeira fase na construção de um sistema de aprendizagem automática e passa pela definição da hipótese.
- **Parâmetros** - Variáveis usadas pelo modelo para tomar decisões.
- ***Learner*** - Parte do sistema que ajusta os parâmetros, comparando a previsão com o resultado obtido.

Depois de definidas estas três partes fundamentais, o sistema segue, por norma, a fase de treino, pela ordem representada na figura, 2.4. Consiste numa fase de treino, seguida de validação e por fim teste. A fase de treino corresponde a um conjunto de exemplos de *inputs* que o modelo irá receber. A fase de validação passa pela avaliação de um conjunto de dados que permite ao modelo ajustar os parâmetros de forma a melhorar os resultados. Por último temos a fase de teste, consiste numa avaliação a ser feita após o treino do modelo para verificar que os resultados são precisos [2].

Figura 2.4: Processo de Treino em Aprendizagem Automática.

2.4.3 Ferramentas

Existem várias ferramentas que ajudam no desenvolvimento de programas de aprendizagem automática. De acordo com [1], estas são as três principais ferramentas de 2020.

- **Scikit Learn**, construída para o desenvolvimento de programas de aprendizagem automática em Python. Disponibiliza uma biblioteca para a linguagem de programação Python. É ideal para análise e *mining* de dados e fornece modelos e algoritmos de classificação, regressão, compartimentação, redução dimensional, seleção de modelo e pré-processamento.

Esta foi a ferramenta escolhida para o desenvolvimento do trabalho pois tem uma boa documentação e esta preparado para classificação em múltiplas categorias.

- **PyTorch**, é baseada em Torch, que é uma ferramenta com suporte de aprendizagem automática baseada em Lua. É fácil de usar e eficiente. Esta ferramenta permite construir redes neuronais, optimizadas, através de "Autograd", isto é, um pacote que oferece uma forma de diferenciação automática. Pode ser usada em plataformas *cloud* pois fornece a possibilidade de treino distribuído da rede. A sua vertente de *front-end* é também um fator diferenciador que permite a utilizadores menos experientes um *feed-back* gráfico.

- **TensorFlow**, oferece uma biblioteca de JavaScript direcionada a aprendizagem automática que permite treinar uma rede neuronal [5].

2.4.4 Elementos Chave

Existem milhares de algoritmos diferentes de aprendizagem automática, mas todos exibem na sua estrutura os seguintes três componentes:

- **Representação**, como se pode representar conhecimento. Inclui, árvores de decisão, redes neuronais, modelos gráficos entre outros.

  - **Árvores de decisão** - Utilizam a estratégia dividir-e-conquistar. Neste tipo de mecânica um problema complexo é decomposto em subproblemas cada vez mais simples. Como exemplificado na figura 2.5, a ideia base das árvores de decisão passa por cada nó, que não seja um nó folha, isto é, um nó terminal, a pergunta que se coloca ao nó tem sempre uma resposta binária, sim ou não. Consoante essa resposta percorremos a árvore até chegar aos nós terminais. Estes nós terminais são os termos, no caso da FPB.

  - **Redes Neuronais** - Uma rede neuronal consiste num conjunto de elementos de processamento, neurónios ou nós, que estão interligados e que representam uma unidade de computação. Cada unidade faz um cálculo com base nos *inputs* que recebe e propaga o resultado para os nós seguintes. Neste tipo de rede, cada nó influencia todos os nós do nível seguinte.

- **Evolução**, no caso de aprendizagem supervisionada, isto é, em que sabemos qual o *output* esperado, existe a necessidade de criar uma função de avaliação do algoritmo. Para assim o algoritmo saber a sua precisão e por consequência melhorá-la. Esta função é denominada função objetivo.

- **Optimização**. Os algoritmos de aprendizagem têm várias variáveis, esta fase faz um estudo, testando várias combinações, de como cada variável pode afetar o resultado. Sendo o objetivo optimizar o resultado final.

2.4.5 Tipos de Aprendizagem

- **Com Supervisão**, também conhecida como supervisão indutiva, nestes casos os dados de treino incluem o *output* desejado. Este é o tipo mais conhecido e estudado tendo em conta que é também o mais simples. Este será o método usado no projeto da FPB.

- **Sem Supervisão**. Ao contrário do primeiro caso, por vezes o programador pode não saber qual o *output* ideal.

- **Supervisão Mista**. Uma mistura entre as duas de cima, sendo que os dados de treino incluem alguns *outputs* desejados.

- **Supervisão Reforçada**. Recompensa sequências de ações, muito usado em inteligência artificial.

Neste projeto são usadas árvores de decisão, em detrimento de redes neuronais. As árvores de decisão, são por norma utilizados em algoritmos de aprendizagem com supervisão. Algumas das vantagens em usar árvores de decisão [15]:

- São um dos melhores algoritmos de seleção de variáveis.

- São rápidos e têm em conta interações entre variáveis.

- Identificam sub-grupos. Cada terminal ou folha intermédia pode ser visto como um sub-grupo da população.

- Lidam bem com valores ausentes.

2.4.6 Amazon Web Services

O serviço responsável pela algoritmia e processamento de dados vai estar hospedado nos servidores da AWS.

Esta solução traz uma série de vantagens, quando comparado com a aquisição de um servidor próprio para hospedar o serviço. Das várias vantagens desta solução destacam-se:

- **Flexibilidade** - A principal vantagem do sistema fornecido pela AWS, é a escalabilidade disponibilizada. Isto não só alivia trabalho de planeamento de uma infraestrutura como, ao mesmo tempo, permite aumentar ou diminuir o número de servidores consoante o tráfego do *website*, permitindo replicar servidores, em casos de tráfego elevado ou diminuir no caso de não ser necessário, poupando assim energia e dinheiro.

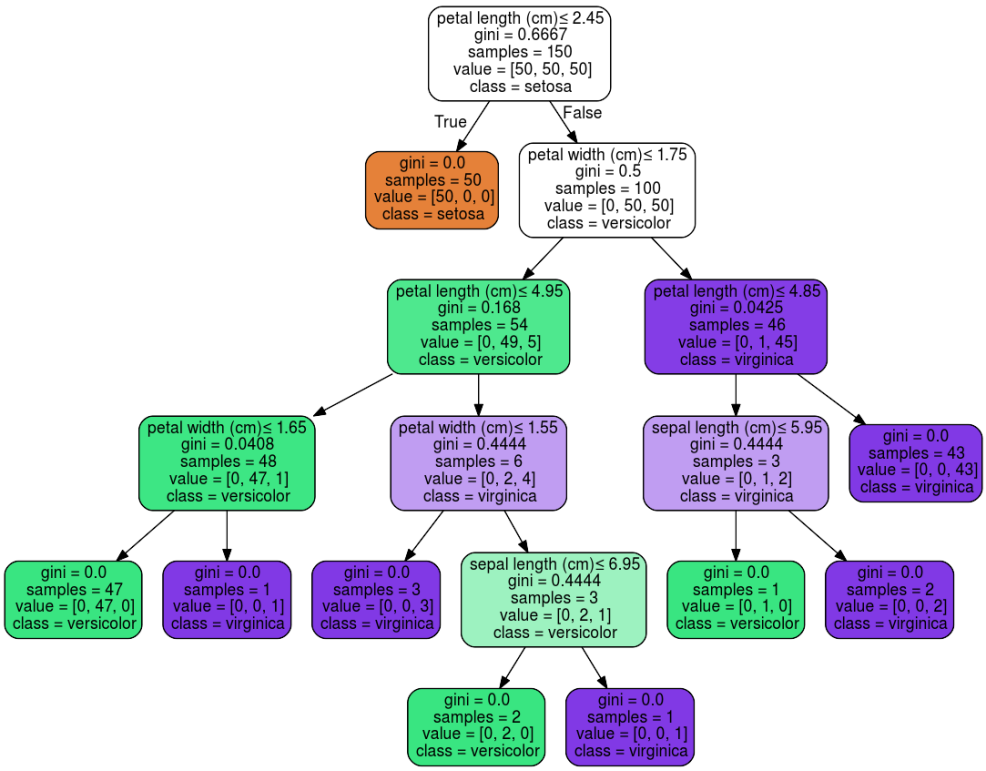


Figura 2.5: Exemplo árvore de decisão.

- **Liberdade** - Outra das grandes vantagens do serviço é a possibilidade de o desativar de um dia para o outro sem quaisquer custos. Isso é completamente impossível no caso de uma infraestrutura fixa, onde mesmo terminando o servido existe sempre equipamento e espaço fixo para vender ou fazer manutenção.

- **Segurança** - Como a maior plataforma de *web services* em *cloud*, a amazon tem uma estrita politica de segurança com uma série de ferramentas montadas que detetam e bloqueiam acessos indesejados ao serviço, o que é uma grande vantagem para quem quer montar um serviço e não tem bases em segurança.

- **Preço** - Apesar de quase todos os serviços serem pagos o preço é uma grande vantagem em comparação com a criação de uma estrutura própria, com tudo o que isso envolve, planificação, compra de espaço, compra de material, segurança, energia, entre outros.

- **Pagar por Uso** - Por último, outra vantagem da AWS é o facto de se pagar por utilização, e não de forma fixa. O que é perfeito para serviços sazonais ou que precisem de mais servidores numa época especifica do ano.

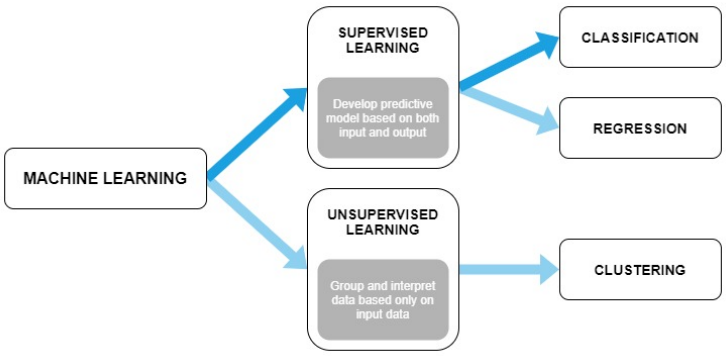


Figura 2.6: Aprendizagem automática esquema.

2.4.7 Conclusão

Neste projeto será usada uma solução classificação, como está representado na figura 2.6, e usaremos a biblioteca de *classification*, presente na plataforma Scikit Learn.

Nesta opção, são associados a um conjunto de dados de *input*, um conjunto de *outputs*:

- **Tipo de Utilizador**
- **Faixa Etária**
- **Género**

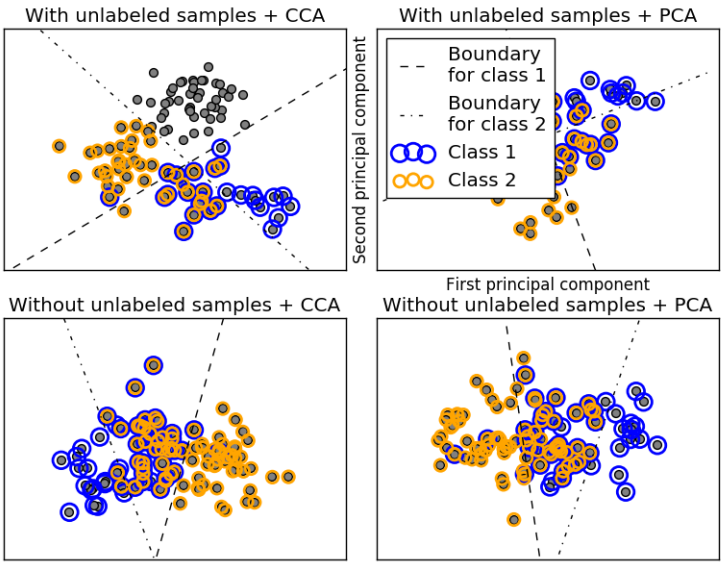


Figura 2.7: Exemplo gráfico de classificação.

2.5 Conclusão

Em suma, é com o objetivo, por parte da FPB, de criar uma ligação mais forte com os seus utilizadores, e deste modo, oferecer-lhes a melhor experiência, que este projeto será desenvolvido. É esperado portanto, que como resultado desta tese, tanto a UX do utilizador, como os indicadores de performance da FPB, sejam optimizados, trazendo assim vantagens para os dois lados. Para os utilizadores, pois o *website*, tentar-se-à aproximar, com a maior precisão possível, dos seus gostos, interesses ou necessidades. E para a FPB, com a melhoria, de indicadores chave, como o tempo de sessão, número de acessos, número de interações e partilhas de conteúdo, entre outros.

Capítulo 3

Trabalho Desenvolvido

Neste capítulo vamos abordar o trabalho desenvolvido, decisões que foram tomadas durante a execução do projeto e a sua justificação.

O trabalho desenvolvido pode ser divido em quatro tarefas principais sendo que o documento seguirá então essa estrutura.

O primeiro capítulo aborda a secção de recolha e tratamento de dados, onde será demonstrado que dados foram recolhidos, de que forma e com que finalidade. O segundo capítulo passa pela fase de definição e descrição dos vários tipos possíveis de utilizadores que podem visitar o *website* da FPB. O capítulo três explica de que forma e com que propósito foram construídos os módulos dinâmicos finais de forma a optimizar a navegação do utilizador pelo *website*. E por último, o quarto capítulo, aborda de que forma poderíamos detetar o utilizador sem que este nos precisasse de dizer que tipo de utilizador é e que conteúdo procura no *website* da FPB.

3.1 Recolha e Tratamento de Dados

Para que seja possível oferecer ao utilizador da FPB a melhor experiência de navegação e utilização do *website* é preciso que a recolha de dados seja criteriosa, eficaz e pouco invasiva para o utilizador. Tendo isso em conta, neste capítulo vamos abordar os vários mecanismos de extração e tratamento de dados que utilizámos de forma a satisfazer essas três propriedades.

Existem duas áreas para a extração de dados, *Client Side* e *Server Side*. O primeiro diz respeito a toda a parte da recolha e tratamento de dados que é processada do lado do

utilizador, isto é, pelo navegador. Por outro lado, o segundo refere-se a toda a parte lógica e de tratamento de dados que é processada do lado do servidor da FPB.

3.1.1 Client Side

Os dados recolhidos no *client side* dizem respeito a toda a recolha ou tratamento de dados que é processada pelo *browser* do utilizador. A proveniência destes dados pode ser dividida em três tipos, os que estão dependentes das ações e do comportamento do utilizador, os que não precisam da intervenção do utilizador e ainda os que exigem uma ação direta por parte do utilizador. Os primeiros resultam da utilização normal do *website*, isto é, fruto das **interações do utilizador**, por exemplo, navegação entre páginas, utilização de filtros ou descarregamento de documentos. Por outro lado, os segundos, resultam de **dados independentes** que não precisam da intervenção do utilizador. Por último existe ainda os dados em que a participação do utilizador é imprescindível, neste caso trata-se de um **formulário** direcionado ao utilizador do *website* da FPB.

3.1.1.1 Interações do Utilizador

As interações do utilizador referem-se a todos os dados que são possíveis retirar de uma utilização normal do *webiste* durante todo o tempo de sessão, que é o tempo desde que o utilizador abre o *website* no navegador até que fecha o navegador.

Para este projeto os dados que decidimos recolher das interações dos utilizadores com o *website* foram os seguintes:

- **Filtros de Pesquisa** - Pelas várias páginas do *webiste* da FPB existe um módulo de filtros que permite filtrar o conteúdo de cada página tendo em conta uma série de categorias previamente associadas a cada conteúdo. Isto permite, por exemplo, que dentro da página de notícias, página esta que lista todas as notícias do *website* da FPB, ao selecionar o filtro de Ano com o valor 2018, a listagem passe a conter apenas as notícias de 2018.

  A ideia passa portanto por manter um mapa de filtros que detete, sempre que o utilizador utiliza um filtro, os seguintes parâmetros: Em que página está o utilizador, em que categoria carregou e qual o valor selecionado.

  Em 3.1, podemos encontrar um exemplo da estrutura criada quando um utilizador seleciona um filtro, sendo que, cada filtro selecionado criará um entrada do género da demonstrada.

  Listagem 3.1: Estrutura após o utilizador selecionar o valor 2018/2019 do filtro Época na página noticias

```
{
"0":{
    "page":"/noticias/",
    "categoryName":"Época",
    "categoryValue":"2018/2019"
    }
}
```

Esta informação é mantida em *Cookies*, para que possa ser reposta mesmo após o fecho do navegador. Para além disso é ainda enviada de 30 em 30 segundos para o *server side* para que possa ser mapeada e utilizada no preenchimento automático de filtros, este tema será abordado, de forma mais detalhada, na secção de *server side*.

No final de uma sessão a recolha desta informação resultará na seguinte estrutura 3.2.

Listagem 3.2: Exemplo final do mapa de filtros

```

{
    "0":{
        "page":"/noticias/",
        "categoryName":"Época",
        "categoryValue":"2018/2019"
    },
    "1":{
        "page":"/noticias/",
        "categoryName":"Género",
        "categoryValue":"Masculino"
    },
    "2":{
        "page":"/noticias/",
        "categoryName":"Escalao",
        "categoryValue":"Sub18"
    },
    "3":{
        "page":"/calendario/",
        "categoryName":"Competiçao",
        "categoryValue":8675
    },
    "4":{
        "page":"/comunicado/",
        "categoryName":"Categoria",
        "categoryValue":" Conselho de Arbitragem"
    },
    "5":{
        "page":"/comunicado/",
```

```
30                  "categoryName":"Época",
31                  "categoryValue":"2018/2019"
32              }
```

- **Partilha** - No *website* da FPB existe um módulo de partilha, em todos os conteúdos que sejam partilháveis, como notícias, fotografias, vídeos, histórias ou opiniões. Como o número de conteúdos partilhados pode ser um critério importante para identificar que tipo de utilizador temos presente no *website* esse valor é adicionado a uma variável e mantido em *cookies* durante toda a sessão do utilizador.

  Este critério pode ser importante já que por norma os utilizadores que partilham mais informação são os adeptos em detrimento dos atletas ou árbitros.

- **Downloads** - Existem ainda, no *website* da FPB, alguns documentos que podem ser descarregados pelo utilizador, estes documentos estão presentes nas páginas de documentação e comunicados. Este valor é, sem dúvida, um critério importante para identificar que tipo de utilizador temos presente no *website*, visto que um utilizador com mais documentos descarregados tem mais probabilidade de ser algum órgão ligado à parte mais regulamentar do desporto, pode ser um árbitro, por exemplo, pode ser um árbitro à procura de uma diretiva em relação aos jogos da próxima semana.

- **Cliques e Navegações** - Por último, adicionámos mais duas variáveis que consideramos relevantes na identificação do utilizador. São elas, o número de cliques totais, efetuados pelo utilizador durante a sessão, e ainda o número de navegações, isto é, a quantidade de vezes que o utilizador mudou de página.

  A variável que controla o número de navegações vai ainda ser importante para a secção "Controlo de Tempo" que falaremos mais à frente no âmbito do tratamento de dados *server side*.
  Estes valores são importantes para conseguir perceber se é um utilizador experiente e conhecedor do *website* que com poucos cliques consegue chegar aos dados que necessita ou um utilizador que anda mais perdido porque, por exemplo, a informação que procura pode ser mais difícil de encontrar, como é o caso de um utilizdor familiar de atleta que procura o jogo do seu filho que joga nos SC Marinhense no escalão Sub-16.

Todas estas variáveis são passadas para *server side* para que possam ser atualizadas na base de dados na entrada referente a esta sessão.

3.1.1.2 Dados Independentes

Dados independentes são dados inerentes à sessão iniciada pelo utilizador, mas que não dependem das suas ações. Ou seja, enquanto no tipo de dados anteriores a extração dos dados esta dependente das interações do utilizador com o *website*, neste tipo de dados isso já não se verifica.

Estes dados consistem em dados de localização, do utilizador, e data no início da sessão.

- **Localização** - Em relação à localização utilizámos uma API disponibilizada pela Google que permite obter a localização com precisão ao nível do concelho em que o utilizador se encontra. A Geocoding API permite, a partir das coordenadas geográficas do utilizador obter informações legíveis para humanos, como em que concelho este ponto geográfico se encontra.

  Retirar a informação sobre em que concelho o utilizador se encontra é importante no caso concreto do *website* da FPB pois existe uma área destinada as associações e esta área funciona como um mini *website* dentro do *website* principal mas com todos os conteúdos filtrados para a associação selecionada. Conseguir detetar qual o concelho do utilizador permitirá a criação ou redirecionamento de conteúdos relativos à associação que ajudarão o utilizador a encontrar os seus artigos de interesse mais rapidamente e com menos navegação.

  A par do que acontecia com os dados retirados das interações com o utilizador também a localização é passada para *server side* e aí guardada na base de dados referente à sessão do utilizador.

- **Data** - Consiste em obter os dados relativos ao dia da semana e hora em que o utilizador iniciou a sessão. Estes dados, apesar de simples, são bastante importantes na identificação dos diferentes tipos de utilizadores, já que diferentes tipos de utilizadores terão dias da semana e horas preferenciais de acesso ao *website* da FPB.

  Este valor pode ser decisivo já que o dia da semana em que o utilizador está a aceder ao *webiste* é muitas vezes indicativo do que este procura. Enquanto árbitros podem tender a aceder a meio da semana como forma de preparação, um familiar de atleta é mais provável aceder Sexta-Feira ou durante o fim de semana para aceder ao calendário e resultados da sua filha, por exemplo.

Formulário de Identificação de Perfil

Para terminar a parte da extração e tratamento de dados que é feita do lado do utilizador existe ainda um formulário. O formulário tem como objetivo categorizar os *inputs* que

serão fornecidos ao algoritmo durante a fase de treino. Para esse efeito é pedido ao utilizador que responda a um mini questionário, com apenas três perguntas e em que a resposta é feita através da seleção de opções definidas em 3.2.2.

Este formulário aparecerá de forma discreta no *website* para não ser incomodativo para o utilizador. Por defeito o formulário está minimizado, apenas com uma pequena pestana visível. Quando o utilizador coloca o rato por cima dessa pestana, o formulário expande ficando totalmente visível. Podemos ver essa transformação na figura 3.1.

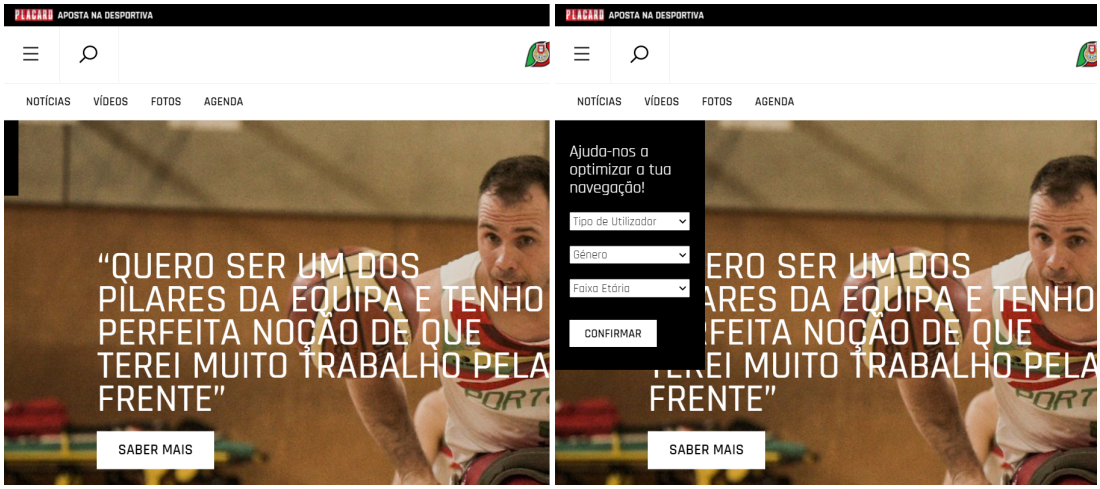


Figura 3.1: Formulário de Identificação de Perfil de fechado para aberto.

As três perguntas feitas ao utilizador nesta fase de treino são as perguntas que, após o algoritmo treinado, fazemos ao próprio algoritmo para obter esta informação, sem precisarmos assim da intervenção do utilizador. Estas perguntas dão-nos a seguinte informação: Qual o tipo do utilizador, qual a sua faixa etária de interesse e qual o seu género de interesse. A primeira é essencial para definir quais dos módulos que foram definidos em 3.3.2 e as outras duas têm influência no conteúdo desses módulos.

3.1.2 Server Side

No *server side*, é feito o **controlo de sessão**, isto é, a inicialização de todos os dados que são mantidos em sessão durante toda a comunicação do utilizador com o *website*.

Existe ainda, no entanto, alguma informação que pode ser extraída, não obtida no *client side*, como o **Controlo de Tempo** que consiste em fazer um controlo do tempo que o utilizador permanece em cada área do *website* da FPB.

Para além destas duas fases existe ainda um tratamento de dados, tanto ao nível do mapeamento dos dados recolhidos através dos filtros, **mapear filtros**, como também a preparação dos dados de forma a inseri-los na base dados, **guardar dados**.

Controlo de Sessão

No início de uma sessão por parte do utilizador é criada do lado do servidor uma série de campos que ajudarão a manter os dados recolhidos em sessão. É gerado portanto um novo *session_id* que é um identificador único da sessão corrente e é esse valor que servirá de chave primária da base de dados, sempre que quisermos aceder ou modificar determinados dados da sessão.

O código referente à inicialização da sessão pode ser visto em 3.3.

Listagem 3.3: Inicio de sessão

```
$_SESSION['tracking'] = array();
$_SESSION['start_time'] = strtotime("now");
session_regenerate_id();

$request = array(
    "requested" => $_SERVER['REQUEST_URI'],
    "timestamp" => $_SERVER['REQUEST_TIME'],
    "referer" => $_SERVER['HTTP_REFERER'],
    "ip" => $_SERVER['REMOTE_ADDR'],
    "timeinpage" => 0
);

$_SESSION['session_tracking_id'] = session_id();
array_push($_SESSION['tracking'], $request);
add_db_request($request);
```

Controlo de Tempo

Após a sessão estar iniciada avançamos para a última tarefa de extração de dados do utilizador e consiste em fazer um controlo do tempo por área do *website* da FPB. O *website* da FPB está dividido nas seguintes áreas:

- **Últimas** - Secção do *website* onde o utilizador tem acesso ás últimas novidades e tem páginas como a página de noticias, agenda, comunicados, documentação, histórias e etc.
- **Competições** - Secção do *website* composta pelas páginas de calendário, resultados, estatísticas e classificações.
- **Associações** - Mini *website* com todas as principais áreas mas filtradas previamente por associação de basquetebol.
- **Clubes** - Mini *website* com todas as principais áreas mas filtradas previamente por clube de basquetebol.

- **Atletas** - Área do *website* onde podemos pesquisar e obter o detalhe de qualquer atleta pertencente à FPB.

- **Sobre Nós** - Área mais institucional do *website* onde podemos encontrar desde páginas onde são apresentados os valores da FPB à página de contactos.

A cada navegação do utilizador é verificado se houve uma mudança de área e se sim, é adicionado o tempo desde da última navegação até este pedido. O valor é alterado de imediato na base de dados para minimizar a perda de informação em caso de queda da sessão.

Durante este processo, ou seja, a cada navegação, é ainda calculado o tempo médio entre navegações, leia-se, o tempo médio em cada página. E é ainda atualizado o tempo total da sessão.

O controlo do tempo em cada área do *website* pode ajudar a identificar áreas de interesse do utilizador e por consequência ajudar a identificar o tipo de utilizador.

Mapear Filtros

Acabada a fase de extração dos dados é necessário tratar esses dados de forma a que possam ser utilizados. O mapeamento dos filtros passa pela passagem da estrutura descrita anteriormente e representada em 3.2 para uma nova estrutura que possa depois ser utilizada para o pré preenchimento de filtros ao longo da sessão. De notar que esta fase é completamente independente e funcionará sempre desde que o utilizador já tenha selecionado filtros durante esta sessão.

A ideia deste mapa é ter uma relação de filtros mais utilizados pelo utilizador para cada categoria. Desse modo sempre que o utilizador entrar numa página que possua o módulo de filtros estes serão preenchidos automaticamente consoante este mapa, este tema será abordado mais à frente na 3.3.2. Para este efeito é portanto construído um mapa em que a chave é a categoria da filtragem, apresentada em 3.2 como *"categoryName"* e para cada campo de categoria existe uma listagem das opções já selecionadas até agora pelo utilizador ordenadas de forma decrescente pelo número de vezes que foram utilizados.

Existem ainda dois valores muito importantes retirados nesta fase, opção de filtro de género e faixa etária mais utilizada. Isto permite identificar, após a fase de treino, ou seja, na altura em que já não existe Formulário de Identificação de Perfil, ajudar a definir estes dois campos que são *outputs* do algoritmo de aprendizagem automática. Estes valores, durante a fase de treino serão iguais à resposta dada pelo utilizador no Formulário de Identificação de Perfil, sendo apenas essenciais numa fase mais adiantada, quando esse formulário for removido.

Guardar Dados

Esta é a última fase da extração e tratamento dos dados e é nesta fase que todos os dados recolhidos são passados para a base de dados. Estes dados são introduzidos na base de dados numa tabela chamada *user logs*. Esta tabela tem como chave primária o identificador de sessão e é constituída pelos vários campos descritos anteriormente e com a seguinte estrutura 3.4. Fica assim concluída a extração e tratamento de dados ficando estes prontos a serem utilizados nas secções seguintes.

Listagem 3.4: Código da criação de User Logs

```
1
2  CREATE TABLE `user_logs` (
3    `session_id` varchar(255) NOT NULL DEFAULT '',
4    `ip_address` varchar(50) NOT NULL DEFAULT '',
5    `time_in_associacoes` int(12) NOT NULL,
6    `time_in_competicao` int(12) NOT NULL,
7    `time_in_competicoes` int(12) NOT NULL,
8    `time_in_sobre_nos` int(12) NOT NULL,
9    `time_in_ultimas` int(12) NOT NULL,
10   `time_in_clubes` int(12) NOT NULL,
11   `time_in_atletas` int(12) NOT NULL,
12   `time_in_outros` int(12) NOT NULL,
13   `n_shares` int(12) NOT NULL,
14   `n_downloads` int(12) NOT NULL,
15   `n_shares` int(12) NOT NULL,
16   `n_clicks` int(12) NOT NULL,
17   `n_navigations` int(12) NOT NULL,
18   `avg_time_in_page` int(12) NOT NULL,
19   `total_time_in_website` int(12) NOT NULL,
20   `session_day` int(1) NOT NULL DEFAULT '0',
21   `session_hour` int(12) NOT NULL,
22   `location` varchar(50) NOT NULL DEFAULT '',
23   `pref_gender` binary(1) NOT NULL DEFAULT 'M',
24   `pref_age_group` varchar(50) NOT NULL DEFAULT '-',
25   `gender` binary(1) NOT NULL DEFAULT 'M',
26   `age_group` varchar(50) NOT NULL DEFAULT '-',
27   `target` varchar(50) NOT NULL DEFAULT '-',
28    PRIMARY KEY (`session_id`)
29 ) ENGINE=MyISAM DEFAULT CHARSET=latin1
```

3.2 Definição de Perfis

Para conseguirmos definir o que seria uma melhor experiência no *website* da FPB, é importante definir o perfil dos utilizadores. Para esse efeito foi necessário fazer um estudo sobre quais os vários tipos de pessoas que visitam o *website*.

3.2.1 Estudo do Perfil

Todos os utilizadores são diferentes entre si mas existem alguns parâmetros que fazem com que as preferências destes utilizadores se aproximem. É nestes parâmetros que nos vamos apoiar para tentar otimizar ao máximo a experiência do utilizador no *website* da FPB. Desse modo foi necessário definir um número finito de tipos de utilizadores, faixas etárias e géneros.

Definir Tipos de Utilizadores

O *website* da FPB, como *website* relacionado com desporto, atinge uma grande parte da população Portuguesa e é utilizado tanto por adeptos como por pessoas dentro do meio do Basquetebol Português para obter informações importantes.

Deste modo, quando definimos os tipos de utilizadores tentámos que estes englobem a grande maioria dos utilizadores do *website* da FPB.

Os perfis definidos foram os seguintes:

- **Atleta** - Caracteriza-se por um interesse na zona de estatísticas de uma competição específica. Este perfil de utilizador tem um maior foco em estatísticas, logo a criação de módulos e resumos estatísticos é importante.
- **Familiar de Atleta** - Caracteriza-se, normalmente, por um interesse na zona de calendário e resultados, por norma, para competições de escalões mais jovens. Para este tipo de utilizador a obtenção da localização é muito importante pois as competições juvenis não são, na sua grande maioria, competições nacionais.
- **Árbitro** - Acessos a documentação, novas directrizes, notícias relativas a árbitros, são as características que definem este tipo de utilizador. Faz sentido criar um módulo que permita o acesso rápido às documentações relativas a árbitros, nomeações e castigos, como também acesso rápido a notícias da categoria "árbitros".
- **Adepto** - Caracteriza-se por um utilizador que segue as notícias, competições, atletas mais mediáticos. Para este tipo de utilizador faz sentido apresentar os atletas que mais se destacam nas principais competições nacionais assim como as classificações do principal campeonato, masculino e feminino.

Definir Faixas Etárias

A faixa etária de interesse do utilizador também é importante para que consigamos direcionar o utilizador para os seus escalões de interesse no *website* da FPB.

As faixas etárias foram definidas para que seja possível fazer uma associação direta a

um conceito de escalão, já existente no *website*. Esse conceito de escalão está dividido nas seguintes categorias:

- **Mini 8 a Mini 12**
- **Sub 13 - Sub 20**
- **Seniores**
- **Veteranos**

Género

Assim como na faixa etária, o género de interesse do utilizador também é bastante relevante para filtrar o conteúdo do *website* da FPB para o utilizador.

No caso da FPB o utilizador pode definir o seu género de interesse como:

- **Masculino**
- **Feminino**

3.2.2 Previsão de Utilização

Apesar de ser difícil prever para cada um dos tipos de utilizador quais as variáveis mais relevantes, foi feita uma previsão que faz a relação entre os valores previstos para cada variável em relação a cada tipo de utilizador. Esta previsão foi feito utilizando um escala de 1 a 4 para cada uma das variáveis, sendo que 1 significa que o utilizador provavelmente terá um valor baixo nessa variável e 4 um valor muito elevado.

Os resultados desta previsão podem ser vistos na figura 3.2 e demonstram que se prever que os tipos de utilizador sejam suficientemente distintos de modo a ser possível a sua deteção.

3.3 Módulos Dinâmicos

Com vista a melhorar a experiência do utilizador criámos ou afetámos alguns módulos do *website*. Esses módulos são construídos com base nos dados já extraídos e na resposta dada pelo utilizador no preenchimento do Formulário de Identificação de Perfil.

Para cada tipo de utilizador existem módulos diferentes que permitem ao utilizador uma navegação mais direta e harmoniosa ao longo do *website*.

Estes módulos dinâmicos podem se distinguir em duas categorias: os módulos de afetação e os de criação. Os módulos de afetação, tal como o nome indica, afetam módulos

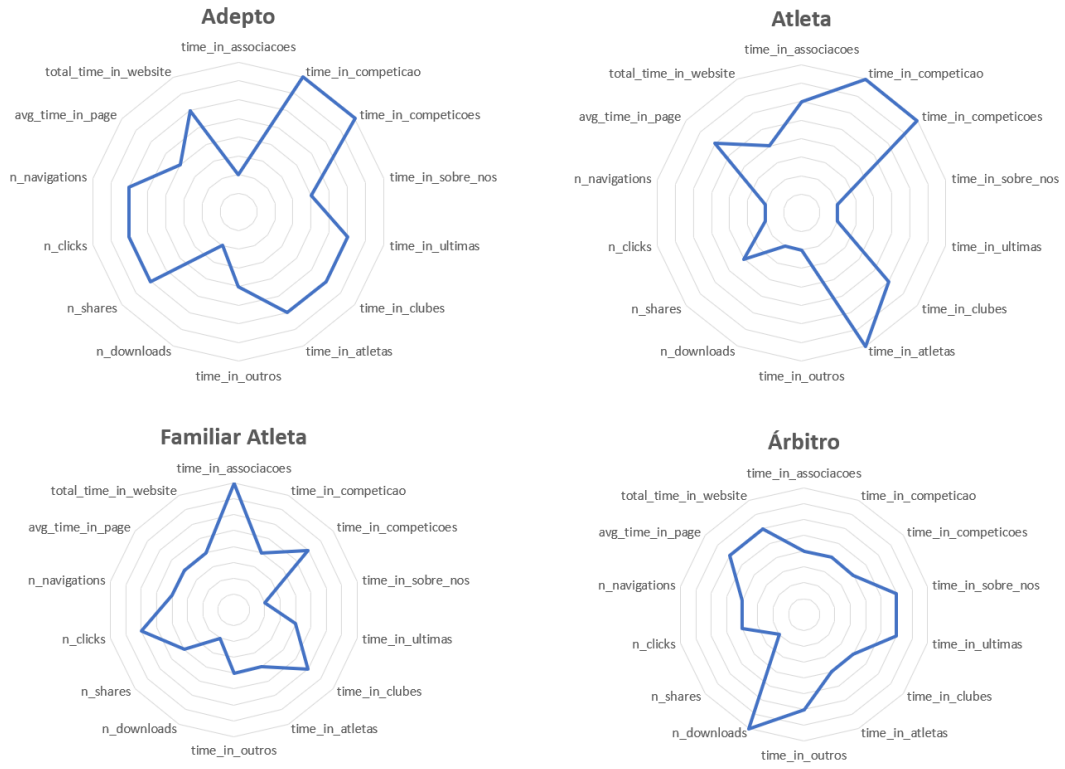


Figura 3.2: Conjunto de gráficos de raios com a previsão da relevância de cada variável no tipo de utilizador.

do *website* já existentes. Por outro lado, os módulos de criação são módulos criados de raiz para permitir uma melhor experiência ao utilizador.

3.3.1 Afetação

Os módulos de afetação iram influênciar módulos já existentes no *website*, ou seja, por exemplo, poderá passar por adicionar um acesso direto à página de "Resultados" da Associação de Basquetebol de Lisboa, após ter sido detetado que o utilizador se encontra em Lisboa e é um familiar de atleta, ou seja, por norma, tem interesse em competições regionais de cariz infantojuvenil. O *website* da FPB, tem por base, na sua construção, o utilizador mais habitual. Ou seja, em todas as páginas reservadas a conteúdo, o *website* está feito para mostrar o conteúdo mais relevante ao nível nacional. No entanto, o *website* da FPB contém uma quantidade muito elevada de informação referente a todos os escalões de basquetebol de todas as associações de Portugal. Se imaginarmos um utilizador que segue uma competição júnior, por exemplo, este utilizador é forçado a uma série de navegações para chegar ao sítio pretendido. Mesmo para um utilizador conhecedor do *website* torna-se cansativo a cada página ter de utilizar vários filtros para chegar aos resultados pretendidos. Deste modo é necessário optimizar estas ações, tendo como objetivo minimizar o acesso a páginas intermédias ou a quantidade de filtros utilizados.

Filtros

O módulo de filtros é constituído por uma série de caixas de selecção e a cada opção selecionada pelo utilizador altera a secção de conteúdos consoante o filtro selecionado. Um exemplo pode ser visto na imagem 3.5.

As páginas de "Notícias", "Documentação" ou "Comunicados", possuem filtros baseados em categorias associadas a cada elemento. Estas categorias foram criadas no *Backoffice* de *WordPress*, do *website*. Isso permitiu associar a cada notícia os seguintes campos:

- **Tipo** - Este é a divisão de mais alto nível do *website*, visto que o divide em três principais áreas, basquetebol 5vs5, que é o basquetebol mais conhecido e praticado, basquetebol 3vs3 e ainda basquetebol de cadeira de rodas (BCR).
- **Categoria** - Filtro composto por uma série de categorias que refletem os principais agentes do desporto. Aqui encontramos categorias como "Associações", "Atletas", "Juízes", "Competições", entre outros.
- **Época** - Divide o conteúdo para que possa ser filtrado por época desportiva.
- **Género** - Divide o conteúdo para que possa ser filtrado por género, masculino ou feminino.
- **Escalão** - Divide o conteúdo para que possa ser filtrado por escalão. Este filtro tem opções desde os escalões infantis, Minis-8 a Minis-12, até aos escalões seniores.
- **Competição** - Caso os filtros "Época", "Escalão" e "Género" estejam selecionados este filtro fica desbloqueado e permite filtrar por uma competição especifica, por exemplo, "XIV Taça Nacional Sub 16 Masculino".
- **Período** - Filtro que permite filtrar o conteudo temporalmente com um sistema de janela de datas, por exemplo, de "19 NOV 2020" a "30 NOV 2020".

Nesta imagem, 3.5, podemos ver, como exemplo, este módulo na página de notícias numa utilização normal, isto é sem filtros pré-selecionados. A ideia é utilizar a informação recolhida, tanto de filtros já selecionados como do Formulário de Identificação de Perfil preenchido pelo utilizador para pré-preencher alguns destes filtros. Na figura 3.4 apresentamos o exemplo de como um utilizador com interesse em basquetebol feminino e no escalão Sub-16 veria a página de calendário.

Áreas Publicitárias

Ao longo do *website* existem também algumas áreas publicitárias. Estas áreas são, nalguns casos, utilizadas para promover produtos da loja oficial da FPB. Na loja é possível adquirir produtos desde de equipamentos, acessórios, bolas entre outros. Seria portanto

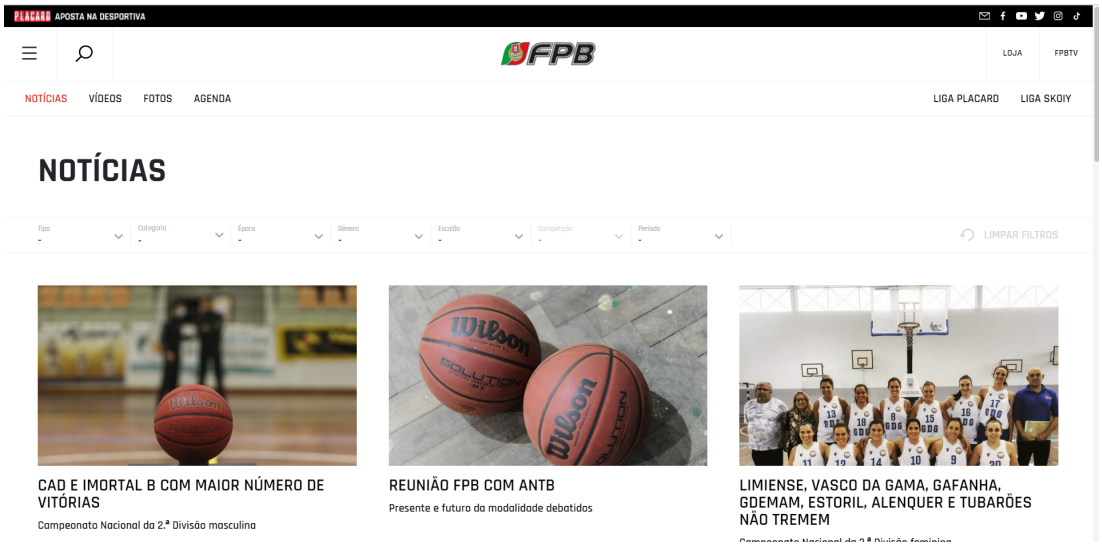


Figura 3.3: Exemplo de filtros sem pré-seleção

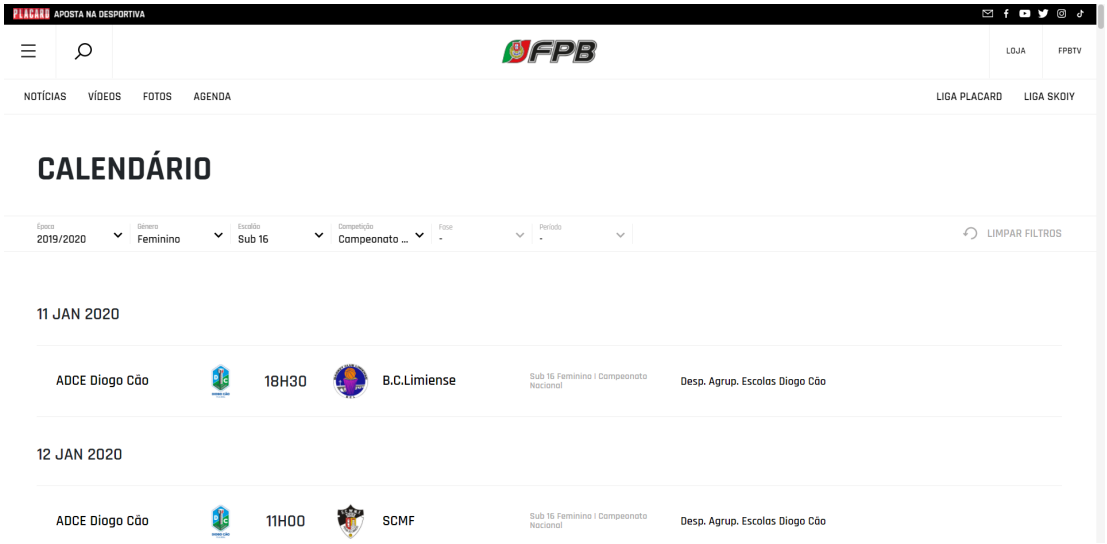


Figura 3.4: Exemplo de filtros pré-selecionados

interessante utilizar a informação já obtida de forma a apresentar nestes espaços publicitários os produtos que podem ser mais relevantes para este utilizador em específico. Este ponto é também particularmente importante para um dos objetivos desta tese que seria a promoção da loja FPB de forma a aumentar as vendas na loja *online*.

3.3.2 Criação

Para além dos módulos de afetação foram também criados de raiz dois módulos. Um deles passa por adicionar uma secção na qual são inseridas **ligações rápidas**, personalizáveis e ajustáveis aos interesses do utilizador em questão. O outro módulo, mais complexo, ao qual chamamos de **"FPB Smart Widget"**, é uma espécie de balão que acompanha o utilizador durante toda a sua navegação no *website* da FPB.

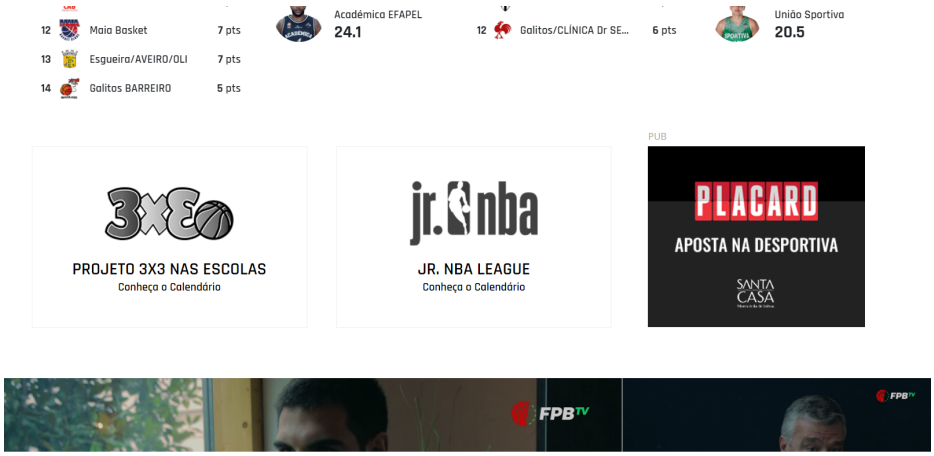


Figura 3.5: Área destinada a publicidade.

Ligações Rápidas

Este módulo passa por adicionar algumas ligações rápidas para benefício e optimização da navegação do utilizador.

O módulo encontra-se no cabeçalho do *website* junto a algumas ligações rápidas comuns a todos os utilizadores.

Como pode ser visto na figura 3.6, as ligações rápidas aparecem a dourado no cabeçalho do *website* e acompanham o utilizador ao longo das várias páginas do *website*. Estas

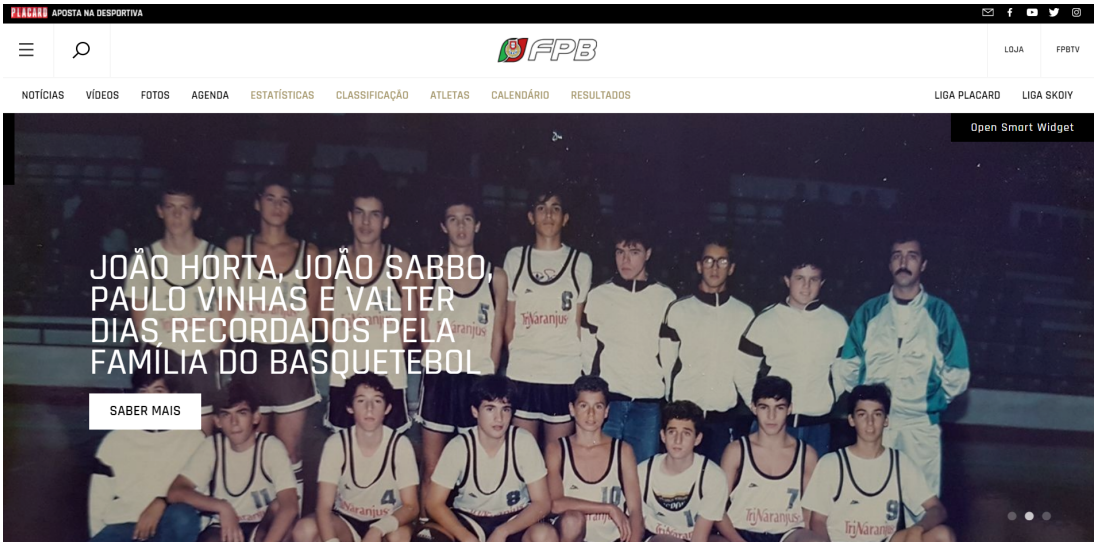


Figura 3.6: Exemplo de ligações rápidas

ligações rápidas podem ser bastante benéficas sendo que evitam cliques que o utilizador

teria de efetuar para chegar à página pretendida.

Com isto em mente definimos então uma série de ligações para cada tipo de utilizador.

- **Adepto** - Para o adepto normal definimos que as ligações mais interessantes para o utilizador são: "Calendário", os "Resultados", a "Classificação" e ainda a página de "Clubes". No caso de ser possível obter a localização devemos então mostrar também a página da associação referente a localização retirada.
- **Atleta** - Para um atleta as ligações que seriam mais interessantes para são as "Estatísticas", a "Classificação", a página de "Atletas", o "Calendário" e ainda os "Resultados". No caso de ser possível obter a localização devemos então mostrar também a página da associação referente a localização retirada.
- **Familiar de Atleta** - Para um familiar de atleta, que por norma tende a ter interesse em competições locais, as ligações dependem bastante de conseguir obter a localização do utilizador visto que é preciso saber que associação o utilizador segue. Tendo isto em conta, e assumindo que obtivemos a localização do utilizador, iremos apresentar atalhos para a página de "Associação", "Calendário Associação" e ainda "Resultados Associação".
- **Árbitro** - Para um árbitro, isto é, perfil de um utilizador que procura informação mais formal em relação a directrizes da FPB, as ligações que definimos foram "Comunicados" e "Documentação", visto que provavelmente é a informação que este tipo de utilizadores procura.

Em suma pretendemos com este módulo optimizar as navegações entre páginas, minimizando a quantidade de interações que o utilizador precisa de fazer para navegar para as páginas necessárias. Isto faz com que o utilizador tenha uma melhor experiência, evitando assim que se perca pelo *website* sem encontrar a informação desejada.

FPB Smart Widget

Ainda mais interessante do que optimizar a navegação do utilizador pelo *website* da FPB é a criação de módulos que permitam o acesso a alguma informação sem que o utilizador tenha de navegar pelo *website*. Um exemplo da utilidade de módulos deste tipo seria um utilizador que costuma entrar no *website* para ver as últimas notícias relacionadas com atletas, as estatísticas mais relevantes de uma determinada competição ou a classificação de uma competição. Se juntarmos toda esta informação que por norma está em páginas distintas, neste caso, respetivamente, nas páginas de "Notícias", "Estatísticas" e "Classificações", podemos oferecer ao utilizador grande parte da informação que procura num só módulo.

Este módulo pode ser acedido com um clique num botão que aparece no canto superior direito em todas as páginas. Após o utilizador abrir o Smart Widget este fica com um comportamento "sticky", o que significa que fica posicionado no canto superior direito mesmo que o utilizador se movimente na página. A qualquer momento o utilizador pode minimizar o módulo para que não cause incómodo na utilização do *website*.

A estrutura do módulo é semelhante entre tipos de utilizador, para uma consistência gráfica. Encontra-se sempre dividida em duas secções, uma secção de fundo branco que pode conter listagens de equipas, carrossel com jogadores ou listagem de documentos. E uma secção de fundo dourado que contém um carrossel de jogadores ou de notícias. Este módulo, no seu conteúdo, terá também diferente informação consoante o tipo de utilizador, pois cada utilizador estará interessado em diferente informação de rápido acesso.

Existem duas formas de obter os dados que vamos apresentar neste módulo. Alguns dados como "Documentos", "Comunicados" ou "Notícias" encontram-se guardados na Base de dados do *Wordpress*, plataforma sobre a qual o *website* da FPB está montado. A outra parte dos dados, onde se incluem, "Estatísticas", "Resultados", "Classificações" ou "Calendário" estão hospedados num serviço externo sendo portanto necessário a API SA para comunicar com este serviço. Foi portanto necessária uma conjunção destas duas formas de obtenção dos dados para ser possível montar os módulos pretendidos.

Segue-se então a definição dos conteúdos que compõem o Smart Widget, para cada tipo de utilizador obtido:

- **Adepto** - Quando na presença de um Adepto, o módulo é direcionado para o conteúdo provavelmente mais procurado no *website* da FPB. Este conteúdo passa por mostrar os melhores jogadores das competições mais populares assim como as classificações dessas competições.
  Um exemplo de uma versão final deste módulo pode ser visto na figura 3.7. Na parte superior do módulo apresentamos os primeiros classificados das duas principais competições nacionais, a liga masculina e a liga feminina. Para obter esta informação utilizamos um pedido feito a API externa que devolve a classificação de uma determinada competição. Precisamos ainda de averiguar qual o tipo da fase da competição, ou seja, se se encontra na fase regular, fase de grupos, *playoffs* ou eliminatórias. Isto porque nas fases regulares existe uma classificação total, isto é, é possível fazer uma comparação entre todas as equipas e ordenar por número de pontos. Nas outras fases a classificação é sempre parcial. Na fase de grupos existe uma classificação por grupo. Na fase de *playoffs* existe uma classificação por fase da competição, ou seja, 1/8 de final tem uma lista de equipas que passaram esta fase, por exemplo. E por último as eliminatórias onde também, o que faz sentido mostrar, é uma lista de equipas não eliminadas.

O exemplo presente na figura 3.7 é o exemplo de uma fase regular, sendo nestes casos mostrados os primeiros quatro classificados da competição.

Na secção de baixo, na figura 3.7 a secção com fundo dourado, construímos um carrossel com os melhores atletas presentes nas duas principais competições, liga masculina e liga feminina. Um carrossel é um módulo que permite, tal como um carrossel, rodar entre vários conteúdos. Isto faz com que seja possível mostrar mais informação num espaço mais curto. Este carrossel é constituído por nove blocos, um por cada estatística medida. Estas estatísticas vêm da API externa e passam por estatísticas de "Lançamentos", "Ressaltos" ou "Assistências" por exemplo. Cada um contém o melhor atleta, de cada uma das ligas, para a estatística indicada no topo da secção. Em relação ao atleta é apresentado o seu nome, clube, uma fotografia de apresentação e ainda o valor da estatística medida. Todo o elemento que envolve o atleta é clicável, o que permite ao utilizador, se assim o desejar, navegar diretamente para a página de detalhe do atleta.

Figura 3.7: Exemplo FPB Smart Widget Adepto

- **Atleta** - O exemplo de uma versão final deste módulo pode ser seguido na figura 3.8. Na parte superior construímos um carrossel com os melhores atletas presentes na principal competição do escalão e género de maior interesse para o utilizador, informação obtida através do Formulário de Identificação de Perfil. Este carrossel é constituído por nove blocos também, um por cada estatística medida. Estas estatísticas vêm da API externa e passam por estatísticas de "Lançamentos", "Ressaltos" ou "Assistências". Cada um contém o melhor atleta para a estatística indicada no topo do módulo. Em relação ao atleta é apresentado o seu nome, clube, uma fotografia de apresentação e ainda o valor da estatística medida. Todo o elemento que envolve o atleta é clicável, o que permite ao utilizador, se assim o desejar, navegar diretamente para a página de detalhe do atleta.

No módulo de baixo, módulo com fundo dourado na figura 3.8, apresentamos as principais notícias, filtradas pela categoria "Atletas" e ainda pelo género e escalão de eleição do utilizador. Esta informação é obtida na base de dados do *website* e podemos encontrar um exemplo da *query* feita a base de dados na listagem 3.5. Este módulo é também ele um carrossel que permite navegar de duas em duas notícias e abrir o detalhe da notícia se assim desejado pelo utilizador.
No código apresentado em 3.5 é de assinalar dois campos relevantes do objeto "$args", o campo "posts_per_page" e o campo "data_query". O primeiro indica que iremos retornar todas as notícias, sem limite de quantidade. O segundo permite limitar a *query* de forma temporal, limitando assim a notícias das últimas duas semanas.

Listagem 3.5: Exemplo código de um pedido à base de dados do Wordpress

```
1   $tax_array = array(
2       'taxonomy' => "categoria_noticias", // the custom vocabulary
3       'field' => 'slug',
4       'terms' => 'atletas' // provide the term slugs
5   );
6
7   $args = array(
8       'post_type' => "noticia",
9       'posts_per_page' => -1,
10      'offset' => 0,
11      'tax_query' => array($tax_array),
12      'post_status' => array('publish'),
13      'date_query' => array(
14          'after' => strtotime( '-2 week' ),
15          'before' => strtotime( 'now' )
16      ),
17      'orderby' => array('date' => 'DESC' )
18  );
19
20  $the_query = new WP_Query($args);
21
22  if ($the_query->have_posts()) {
23      while ($the_query->have_posts()) {
24          $the_query->the_post();
25          print_noticia();
26      }
27  }
```

- **Familiar de Atleta** - O módulo para um familiar de atleta é direcionado para as associações. Como grande parte dos familiares de atletas acedem ao *website* para

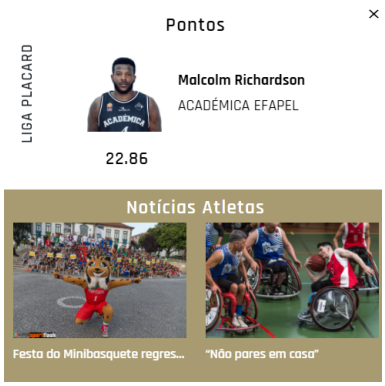


Figura 3.8: Exemplo FPB Smart Widget Atleta

acompanhar as competições dos seus filhos, que provavelmente participam em competições de iniciados ou juvenis que são, por norma, competições menos mediáticas. Um exemplo de uma versão final deste módulo pode ser visto na figura 3.9. Neste exemplo, estamos a simular um utilizador com interesse no basquetebol feminino e no escalão Sub 16. Na parte superior do módulo apresentamos os primeiros classificados das duas principais competições com este género e escalão. A escolha destas competições é apoiada numa tabela de competições existente na base de dados que possui um campo, "prioridade" que permite definir quais as competições mais importantes. Após obter as duas competições prioritárias utilizamos um pedido feito a API externa que devolve a classificação de cada uma dessas competição. Após obter a resposta da API é usado o mesmo mecanismo utilizado, e já explicado, no módulo de "Adepto".

Na secção de baixo, módulo com fundo dourado na figura 3.9, apresentamos as principais notícias, filtradas pela categoria "Associações" e ainda pelo género e escalão de eleição do utilizador. Esta informação é obtida na base de dados do *website* e é bastante semelhante ao pedido referido em 3.5. Sendo que neste caso no objeto "$tax_array" os termos passam a ser "associacoes". Este módulo é também ele um carrossel que permite navegar de duas em duas notícias e abrir o detalhe da notícia se assim desejado pelo utilizador.

- **Árbitro** - Quando na presença de um Árbitro o módulo será optimizado com conteúdos relativos a "Documentação" e "Comunicados", assim como "Notícias" relacionadas com Arbitragem.

  Um exemplo de uma versão final deste modo pode ser visto na figura 3.10. Na parte superior do módulo apresentamos uma listagem de uma mistura de documentos ou comunicados passíveis de descarregar. Cada elemento destes possui o nome ao lado de um botão de descarregamento. Os documentos são ordenados por

Figura 3.9: Exemplo FPB Smart Widget Familiar de Atleta

data, o que faz com que o utilizador consiga detetar rapidamente se saiu um novo documento que necessita de descarregar.
Na secção de baixo, módulo com fundo dourado na figura 3.10, apresentamos as principais notícias, filtradas pela categoria "Árbitros" e ainda pelo género e escalão de eleição do utilizador. Esta informação é obtida na base de dados do *website* e é bastante semelhante ao pedido referido em 3.5. Sendo que neste caso no objeto "$tax_array" os termos passam a ser "juizes". Este módulo é também ele um carrossel que permite navegar de duas em duas notícias e abrir o detalhe da notícia se assim desejado pelo utilizador.

Neste caso mostramos todas as notícias, sem limite de quantidade mas limitadas de forma temporal, limitando assim a notícias das últimas duas semanas.

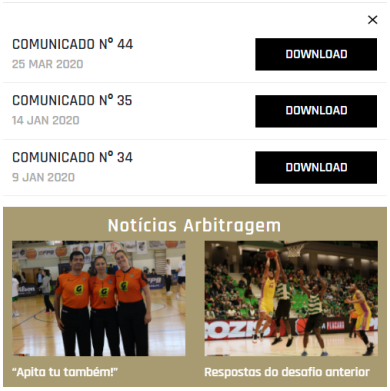


Figura 3.10: Exemplo FPB Smart Widget Árbitro

3.4 Deteção do Tipo de Utilizador

Até agora já definimos o perfil dos utilizadores do *website* da FPB e utilizamos várias estratégias para extrair e tratar os dados do utilizador que nos permitem, na prática, afetar

positivamente a experiência do utilizador. No entanto, de modo a não ser necessária a intervenção do utilizador no preenchimento do Formulário de Identificação de Perfil, descrito em 3.1, foi desenvolvido um algoritmo de aprendizagem automática. Este algoritmo irá permitir que esse formulário seja necessário apenas na fase de treino do algoritmo. Após essa fase de treino deixamos de precisar da interação direta do utilizador para conseguirmos detetar o seu perfil, faixa etária e género preferenciais. A construção deste algoritmo pode ser dividida em quatro partes principais, a **implementação**, a **utilização**, a **fase de treino** e, por fim, a **fase final**.

3.4.1 Implementação

Vamos portanto agora abordar todos os passos que foram necessários para a construção deste algoritmo, mais concretamente, a fase de estudo, a construção do algoritmo e a passagem do algoritmo para um servidor externo da AWS.

Estudo de Algoritmos

Foi necessário no início uma fase de investigação que permitisse responder a algumas questões. Era necessário encontrar o tipo de algoritmo de aprendizagem automática que permitisse resolver este problema em concreto. O problema passa por através de uma série de interações do utilizador com o *website* conseguir obter três campos importantes, o tipo de utilizador, o género de interesse e a faixa etária de interesse. Estes campos coincidem claro com os campos que antes eram respondidos pelo Formulário de Identificação de Perfil.

Após alguma investigação concluímos que o algoritmo ideal para a solução deste tipo de problemas seria um algoritmo de classificação. Existem algumas indicações que um problema seria um problema de classificação ou regressão. Uma vez que existe a possibilidade de, na fase de treino, obter os dados classificados, através do Formulário de Identificação de Perfil. Isto indica-nos de imediato que podemos resolver este problema usando um método de aprendizagem com supervisão. A escolha seria portanto entre o modelo de classificação ou regressão. Entre estes dois modelos o de classificação é o usado nestes casos visto que o *output* do algoritmo será uma ou várias, neste caso três (tipo de utilizador, género e faixa etária), variáveis discretas. Por outro lado o algoritmo de regressão é usado quando o *output* é uma variável continua.

A biblioteca utilizada para a construção do algoritmo foi a *scikit-learn*. Esta é uma biblioteca para Python, de uso gratuito, que suporta vários algoritmos de classificação, regressão ou agrupamento. Precisamente devido ao facto da biblioteca suportar algumas variantes de algoritmos de classificação, foi necessário tomar uma decisão sobre qual o algoritmo que iríamos adoptar.

- **Hipóteses** - De entre as várias funções oferecidas pela biblioteca *scikit-learn*, após investigação, foram escolhidos como principais hipóteses as seguintes funções:
  - LogisticRegression
  - LinearDiscriminantAnalysis
  - KNeighborsClassifier
  - DecisionTreeClassifier
  - GaussianNB
  - SVC
- **Teste de Precisão** - Para decidir qual destes algoritmos iríamos utilizar, optamos por criar um pequeno programa, um teste de precisão, que simula qual a precisão de cada algoritmo para um determinado conjunto de dados.

  Para este efeito foi portanto necessário simular utilizações de vários tipos de utilizador para criar um conjunto de cerca de trinta dados classificados, isto é, dados em que a resposta que pretendemos no futuro ter como *output* do algoritmo, é preenchida pelo utilizador, através do Formulário de Identificação de Perfil.

Listagem 3.6: Código principal do algoritmo de teste de precisão

```
1  models = []
2  models.append(('LR', LogisticRegression(solver='liblinear', multi_class='
       ↪ ovr')))
3  models.append(('LDA', LinearDiscriminantAnalysis()))
4  models.append(('KNN', KNeighborsClassifier()))
5  models.append(('CART', DecisionTreeClassifier()))
6  models.append(('NB', GaussianNB()))
7  models.append(('SVM', SVC(gamma='auto')))
8  # evaluate each model in turn
9  results = []
10 names = []
11 for name, model in models:
12   kfold = StratifiedKFold(n_splits=5, random_state=1, shuffle=True)
13   cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring
       ↪ ='accuracy')
14   results.append(cv_results)
15   names.append(name)
16   print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))
```

- **Análise de Resultados** - Após a análise dos resultados obtidos através do algoritmo de teste de precisão concluímos que o algoritmo com maior taxa de precisão, de entre os algoritmos testados, foi o "DecisionTreeClassifier" e foi por isso com esse algoritmo que avançamos para a construção do algoritmo final.

Construção do Algoritmo

A construção do algoritmo final passou, numa fase inicial, pelo estudo das opções dadas pelo *scikit-learn* para a função "DecisionTreeClassifier".

O algoritmo passa por vários passos. O primeiro passo é a leitura dos dados de teste. Estes dados, de utilização do *website* da FPB, são previamente extraídos da base de dados em Comma Separated Value(s) (CSV). O algoritmo começa por criar um *array* de nomes, nomes estes que correspondem aos campos da tabela da base de dados. Após ter esse *array* criado é utilizada uma função "read_csv" que passa os dados do CSV para uma matriz. Este algoritmo, como é uma árvore de decisão, precisa que todos os campos sejam valores numéricos. No entanto, existem dados, guardados na nossa base de dados, que não são guardados com valor numérico. São eles a localização, o género, a faixa etária e o tipo de utilizador. Para resolver este problema é portanto necessário criar uma variável do tipo "LabelEncoder" para cada um destes campos. Esta variável possui um método, "fit_transform" que permite fazer essa transformação, de campo de texto para valor numérico. Essa variável é guardada de forma a que posteriormente o processo possa ser revertido e consigamos passar do valor numérico de novo para o valor inicial.

Neste momento existem duas opções, uma caso estejamos na fase de treino do algoritmo e outra na fase final.
Na fase de treino do algoritmo, que pode ser seguida em 3.7, é utilizada uma função, "train_test_split" para preparar os dados no formato necessário para serem posteriormente processados. Em seguida é criado o objeto "DecisionTreeClassifier" e chamada de seguida a função "fit", que possui como argumentos os dados preparados pelo "train_test_split".

Listagem 3.7: Código do algoritmo em fase de treino

```
1           X = data[feature_cols] # Features
2       y = np.column_stack((data.gender, data.age_group, data.target)) #[data.gender,
            ↪ data.age_group, data.target] # Target variable
3
4       X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
5
6       # Create Decision Tree classifer object
7       clf = DecisionTreeClassifier()
8
9       # Train Decision Tree Classifer
10      clf = clf.fit(X_train, y_train)
```

Listagem 3.8: Código do algoritmo em final

```
1           input = self.path.split("/?input=")[1]
2           input_csv = StringIO(input)
```

```
3           df = pd.read_csv(input_csv, sep=',', header=None, names=feature_cols)
4           df["pref_age"] = le_pref_age.fit_transform(df["pre_age"])
5           df["pref_gender"] = le_pref_gender.fit_transform(df["pref_gender"])
6           df["location"] = le_location.fit_transform(df["location"])
7           prediction = clf.predict(df.values)[0]
8           gender = le_gender.inverse_transform([prediction[0]])
9           age = le_age.inverse_transform([prediction[1]])
10          target = le.inverse_transform([prediction[2]])
```

Na fase final do algoritmo, que pode ser vista em 3.8, é lido o parâmetro *input*, enviado na *query* do pedido. Este parâmetro é composto por todos os dados recolhidos e guardados na base de dados, só com o valor, e separados por virgulas. Nestes dados já não precisamos de ter os três campos, tipo de utilizador, género e faixa etária, uma vez que estes são os campos de *output* do algoritmo. Para isso é usado a função "predict()" da classe "DecisionTreeClassifier()". Este método prevê quais os três campos de *output* com base nas conclusões calculadas durante a fase de treino em junção com todos os campos fornecidos nesta fase.

Passagem para ambiente AWS

Neste momento já construímos a ferramenta de aprendizagem automática, no entanto esta encontra-se a correr apenas localmente. De forma a aliviar o peso do servidor atual do *website* da FPB, decidimos passar esta ferramenta para um servidor externo, hospedado na AWS.
A AWS permite retirar peso do servidor da FPB, ao mesmo tempo que oferece opções de escalabilidade muito interessantes ao nível da utilização de recursos, permitindo aumentar a capacidade da máquina com base nas necessidades a cada momento.

- **Instalação da Máquina e Bibliotecas** - A primeira fase deste processo passa por aceder a "console.aws.amazon.com", *website* de controlo das várias máquinas criadas por cada utilizador. Cada máquina é chamada de instância. O primeiro passo, após criação de uma conta na AWS, é a inicialização de uma instância.
  A AWS oferece uma grande variedade de tipo de instâncias que variam em termos de sistema operativo, capacidade de armazenamento, aplicações já instaladas na instância, ou preço. Neste caso, e porque ainda se trata de uma fase de testes, a instância criada foi "Amazon Linux 2 AMI (HVM), SSD Volume Type" que pertence à categoria de instâncias gratuitas. Esta é uma máquina muito simples que corre em Linux e não possui interface gráfica. Toda a comunicação teve de ser feita utilizando o "PuTTY" que permite ter acesso ao terminal da máquina em questão. Após ter acesso ao terminal foi necessário instalar todas as bibliotecas para que a ferramenta de aprendizagem automática conseguisse ser interpretada.
  A linguagem utilizada para escrever o algoritmo de aprendizagem automática foi

Python, por isso, foi necessário, instalar o Python3 na máquina. Para além do Python3 foi ainda necessário importar as bibliotecas do "scikit-learn". Para a passagem do código da ferramenta para a máquina na AWS utilizamos o Filezilla que permite facilmente, graças a sua interface gráfica, a passagem de ficheiros da máquina local para a máquina hospedada na AWS.

- **Criação de serviço REST** - Para que o *website* consiga comunicar com a ferramenta de aprendizagem automática é necessário que esta tenha um serviço a correr à espera de pedidos. Para esse efeito foi criado um programa simples, "SimpleHTTPRequestHandler", para que a ferramenta de aprendizagem automática fique sempre atenta a pedidos feitos pelo *website* da FPB. Este bocado de código está representado em 3.9. O serviço fica à espera de comunicações na porta 8000 e a comunicação deverá ser feita através de um método GET HTTP com um parâmetro, chamado "input" onde são passados os dados necessários para que o algoritmo de aprendizagem consiga dar os três campos de resposta que são pretendidos: tipo de utilizador, faixa etária e género.

Listagem 3.9: Código do comportamento REST

```
1   PORT = 8000
2
3   class SimpleHTTPRequestHandler(BaseHTTPRequestHandler):
4
5       def do_GET(self):
6           // O algoritmo fica aqui
7
8       try:
9           server = socketserver.TCPServer(("", PORT),
                ↪ SimpleHTTPRequestHandler)
10          # Activate the server; this will keep running until you
11          # interrupt the program with Ctrl-C
12          print("serving at port", PORT)
13          server.serve_forever()
14      except:
15          pass
16      finally:
17          server.close()
```

- **Testes e Conclusão** - Por fim foi necessário uma fase de teste para verificar se todo o processo se encontrava operacional. Foi necessário por último abrir as portas da máquina presente na AWS para que fosse possível fazer pedidos de qualquer máquina.
  Um exemplo de um pedido final feito à instância montada na AWS terá a seguinte estrutura: "http://3.10.143.14:8000/?input=350,34,0,12,40,22,0,Lisboa,4,1,3,M,Sub16"

e terá como resposta os três campos já referidos. Os valores,separados por virgulas, no exemplo de *input*, corresponde aos campos guardados na base de dados e passados para o algoritmo na fase de teste.

3.4.2 Utilização

Com a ferramenta de aprendizagem automática construída, esta fica assim pronta para a próxima fase do projeto, que passará pela fase de treino e depois disso a fase final. Iremos agora, então, desenvolver em que consistem estas duas fases e a sua importância no resultado final.

Fase de Treino

Numa ferramenta de aprendizagem automática, a fase de treino do algoritmo é uma fase muito importante na veracidade dos resultados finais. É nesta fase, em que o algoritmo recebe, em conjunto com todos os *inputs*, a categorização desses mesmos *inputs*. Isto é, nesta fase o algoritmo aprende de que forma cada padrão de *inputs* estão relacionados com os *outputs* pretendidos.

Para obter dados para esta fase de treino foram feitas algumas simulações de utilização do *website* para cada um dos tipos de utilizador que nos propomos a identificar. Por exemplo, para simular um utilizador do tipo "Árbitro" era feita uma navegação com tendência para páginas de documentos ou comunicados, descarregados alguns ficheiros.

Para esta fase funcionar de forma ideal, a quantidade de dados categorizados deve ser bastante elevada para que o algoritmo não tire conclusões erradas devido à falta de *inputs* que definem um determinado *output*. Devido a alguns imprevistos, relacionados com a pandemia que atravessamos e que afetou bastante o mundo do desporto, nomeadamente desportos de pavilhão. Isso fez com que o campeonato tenha arrancado apenas com as principais competições ao nível nacional, sendo que todas as outras competições foram canceladas. Esta fase ficou, portanto, algo comprometida devido à falta de dados de utilizadores reais, isto é, utilizadores sem ligação ao projeto. Estas questões serão abordadas na conclusão deste capítulo.

Fase Final

Nesta fase, o Formulário de Identificação de Perfil é removido. Todos os dados, referentes à natural navegação do utilizador pelo *website*, continuam a ser extraídos. O objetivo aqui foi fazer navegações semelhantes às que haviam sido feitos na fase de teste e confirmar que o algoritmo conseguia responder de forma correta a qual o tipo do utilizador presente no *website*.

Como o volume de dados era pequeno e, por isso, com poucas alternâncias, os resultados obtidos foram resultados com 100% de precisão do algoritmo. Nesta fase seriam feitos os ajustes aos parâmetros do algoritmo de forma a melhorar a eficácia do mesmo. No entanto, devido à falta de dados esta fase de ajuste aos parâmetros do algoritmo ficou comprometida. Os parâmetros disponibilizados pelo "sickit learn" para ajustar o algoritmo são os seguintes:

- criterion - A função que mede a qualidade de uma divisão da árvore de decisão. Existem dois valores possíveis para este parâmetro "gini" para *gini impurity* e "entropy" para *information gain*.
- splitter - A função que define qual a estratégia de divisão da árvore de decisão, "best" para escolher a melhor divisão ou "random" para escolher uma divisão aleatória.
- max_depth - Valor que define a profundidade máxima da árvore de decisão. Se nenhum valor for passado então a árvore cresce até todas as folhas serem puras, isto é, não precisarem de serem divididas ou até todas as folhas conterem menos de min_samples_split amostras.
- min_samples_split - Número mínimo de amostras necessárias para que o nó seja dividido, por defeito este valor é de duas amostras.
- min_samples_leaf - Número mínimo de amostras necessárias para ser um nó folha, ou seja, ou nó terminal. Uma divisão só será considerada se deixar pelo menos min_samples_leaf para cada uma das suas ramificações, esquerda e direita. Por defeito o valor é 1.
- min_weight_fraction_leaf - A fração de peso mínima da soma do total de pesos de, todas as amostras de *input*, necessária para ser considerado um nó folha. As amostras tem todas o mesmo peso quando sample_weight não é fornecido. O valor por defeito é 0.
- max_features - Valor usado para limitar a procura pela melhor divisão de um nó. Por defeito não existe limitação, no entanto é possível limitar com as seguintes opções:"auto", "sqrt" ou "log2".
- random_state - Controla a aleatoriedade do algoritmo. De forma a não ficar preso em máximos locais, isto é, divisões que o algoritmo acha que são as melhores porque não existem melhores perto dessa, se o critério para a divisão não mudar durante várias *runs*, é selecionada uma aleatoriamente. Por defeito esta opção não está ligada.
- max_leaf_nodes - Número máximo de nós folha permitidos na árvore de decisão.
- class_weight - Utilizado caso se queria dar diferentes pesos as diferentes variáveis de *input*.

- ccp_alpha - Parâmetro de complexidade usado no "Minimal Cost-Complexity Pruning". A sub-árvore com maior custo de complexidade que seja menor que ccp_alpha será escolhida. Por defeito não existe corte da árvore de decisão.

3.5 Conclusão

Em suma, o desenvolvimento dos módulos com diferentes componentes, dependendo do perfil do utilizador constitui um grande passo no caminho do objetivo principal do *website* da FPB e, consequentemente, desta tese, que é uma melhoria na experiência dos utilizadores do *website* da FPB. Para que estes módulos tenham relevância e se adequém a cada tipo de utilizador, dos que foram definidos, foi necessário certificar que todos os dados foram extraídos e preparados de forma eficiente e pertinente sem ser intrusivo para a normal navegação do utilizador. O único fator desviante da normal utilização do utilizador é a necessidade de preenchimento do Formulário de Identificação de Perfil, fator este que foi colmatado com a última fase desta tese, permitindo assim que esse valor, outrora preenchido pelo utilizador, passasse agora a ser calculado pelo algoritmo de aprendizagem automática. No entanto, e apesar de todo o processo estar funcional, devido à pandemia que estamos a atravessar todo o projeto foi atrasando e o novo *website* da FPB acabou por ser lançado com alguns meses de atraso e apenas com as principais competições nacionais, já que os desportos *indoor* voltaram com várias limitações. Contudo, e assim que seja possível incluir este projeto no novo *website* da FPB, tudo estará pronto para que esta fase possa ser retomada, garantindo assim a eficácia do algoritmo.

Capítulo 4

Discussão e Conclusão

Neste capítulo vamos resumir, o que foi feito em termos de produto final e que objetivos propostos foram alcançados. Iremos ainda abordar o que correu menos bem, assim como algum ponto a melhorar e referir ainda de que forma esta ferramenta se pode estender no futuro.

4.1 Produto Final

O produto final obtido é constituído por duas partes, uma parte que de facto impacta o utilizador do *website* da FPB, ou seja, o conjunto de todos os módulos gráficos dinâmicos que foram construídos assim como toda a afetação de módulos já existentes no *website*. A segunda parte é composta por toda a camada de extração e tratamento, assim como a ferramenta de aprendizagem automática que apesar de não serem visíveis ao utilizador constituem uma grande parte deste projeto.

Para tornar esta ideia clara e concreta, vamos seguir um caso específico, em que o utilizador é um "Atleta".
Assim que o utilizador entra no *website* depara-se com a versão base do *website*, figura 4.1.

De seguida dirige-se à página de "Estatísticas" e seleciona os seguintes filtros: "Género - Masculino", "Escalão - Seniores" e "Competição - Taça de Portugal", figura 4.2, e durante uns minutos analisa as estatísticas da competição em causa.

De seguida navega para a página de "Classificação". Quando entra nesta página conseguimos detetar o primeiro ponto de melhoria e otimização da navegação do utilizador, uma vez que este utilizador já preencheu filtros em "Estatísticas", os filtros preenchidos em

Figura 4.1: Exemplo prático, passo 1.

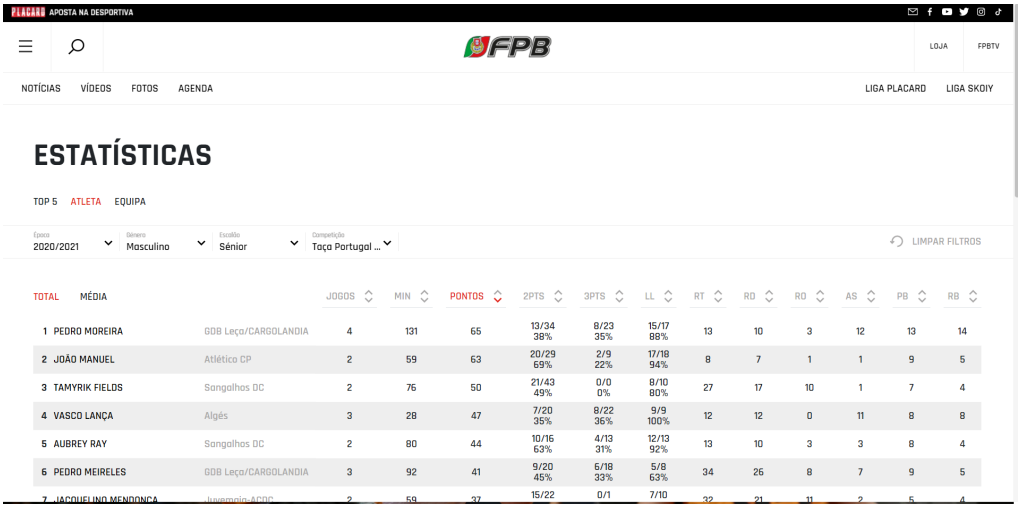


Figura 4.2: Exemplo prático, passo 2.

estatísticas, género, escalão e competição, são automaticamente preenchidos, figura 4.3.

Após 40 segundos no *website* é disparado o primeiro pedido ao algoritmo de aprendizagem automática que com a informação já recolhida irá tentar avaliar que tipo de utilizador está presente no *website*. A resposta chega ao *client side*, recebemos a informação que estamos na presença de um atleta, com interesse no género "Masculino" e na faixa etária "Seniores". Essa informação é enviada para *server side* e é accionado o mecanismo programado na presença de um "Atleta" com as variáveis de interesse "Masculino" e "Seniores". Isso faz com que na próxima navegação do utilizador já estejam presentes tanto as novas ligações rápidas, específicas para atletas, adicionadas à cabeça do *website*, como o módulo "Smart Widget" com informação relevante para um utilizador do tipo "Atleta", figura 4.4.

O processo de identificação do perfil não termina na primeira iteração, já que é possível

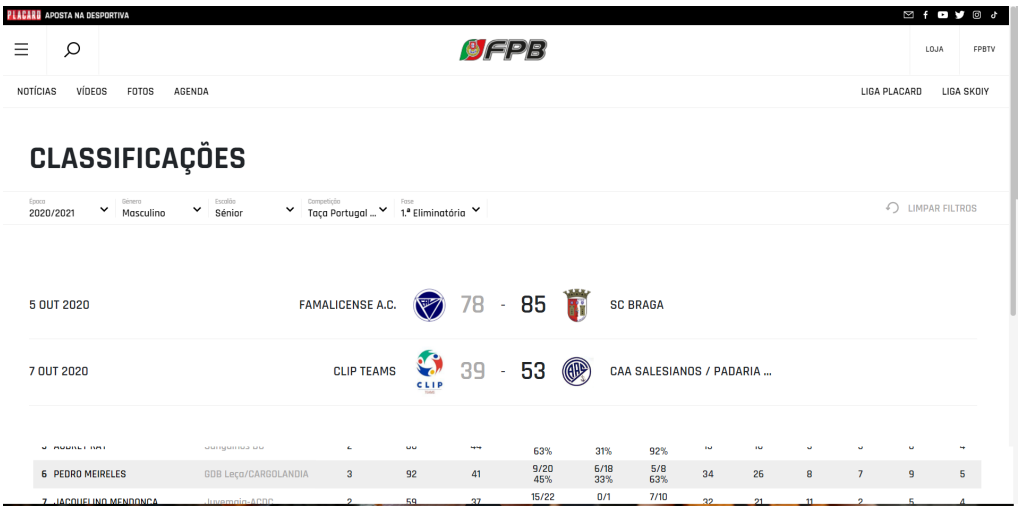


Figura 4.3: Exemplo prático, passo 3.

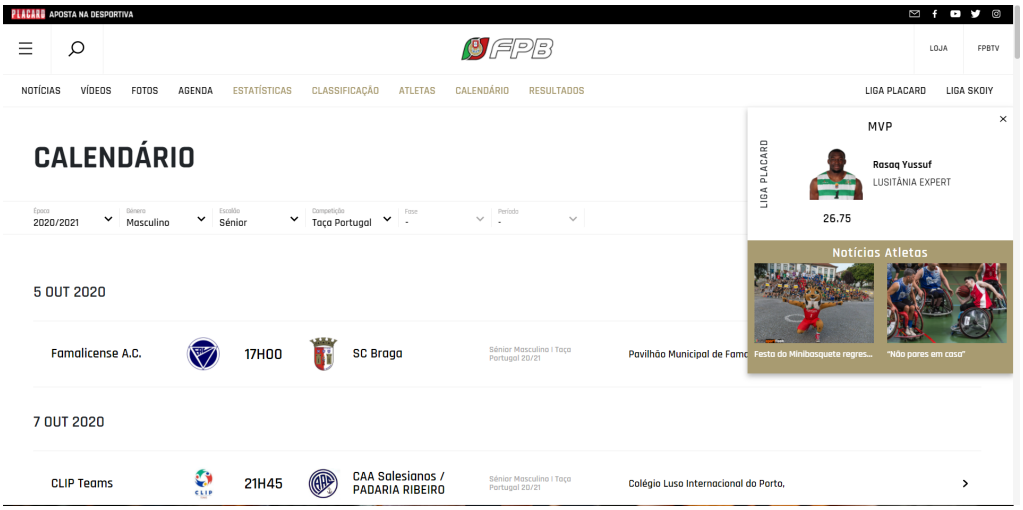


Figura 4.4: Exemplo prático, passo 4.

que por escassez de informação, visto que o utilizador entrou há pouco tempo no *website* e o cálculo do algoritmo possa não estar correto. Desse modo é essencial que de 1 em 1 minuto seja enviado novo pedido ao algoritmo de aprendizagem automática para confirmar o perfil do utilizador e, por consequência, alterar os módulos apresentados.

Todos estes passos levam a que o principal objetivo do trabalho tenha sido atingido. Com a utilização deste mecanismo temos a certeza que a experiência de um utilizador do *website* da FPB sairá beneficiada com todos este módulos que fazem com que a experiência final ao navegar no *website* seja mais personalizada e apta a responder às necessidades de cada um dos tipos de utilizadores.
Em relação aos objetivos que dizem respeito ao aumento dos visitantes do *webiste* da FPB assim como às vendas da loja FPB não podemos tirar nenhuma conclusão, uma vez que o produto ainda não foi adicionado ao atual *website* da FPB. Podemos no entanto assumir

que com esta melhoria tão evidente, no que diz respeito à otimização da navegação para cada tipo de utilizador, esses valores só poderão melhorar.

4.2 Trabalho Futuro

Existem dois principais pontos de melhorias adicionais que podem ser feitas no futuro. Criação de novos perfis e afinação do algoritmo de aprendizagem automática.

Com o passar do tempo pode ser identificada a necessidade de detetar um novo perfil de utilizador. Apesar da solução construída ser flexível, é necessário desenvolver os novos módulos gráficos, assim como pensar se as variáveis até agora extraídas chegam para detetar este novo tipo de utilizador ou se, por outro lado, é necessário adicionar alguma variável. No caso de serem necessárias novas variáveis, exigem-se mudanças tanto no algoritmo, adicionando essas variáveis de *input*, como na própria extração e armazenamento na base de dados.
Em relação à afinação do algoritmo de aprendizagem automática, é um ponto importante e que arrancará assim que seja possível inserir o produto no novo *website* da FPB, pois só assim será possível obter dados reais de navegação no *website*. Isto deverá ser possível assim que a situação no mundo do desporto voltar à normalidade e deixar de ser afetada com todas as restrições atualmente presentes devido à COVID-19.

Chegamos portanto ao fim com a certeza de que o produto final é um produto relevante, útil e que vai melhorar a experiência de todos os utilizadores do *website* da FPB.

Bibliografia

[1] *11 Most Popular Machine Learning Software Tools In 2020*. 2019. URL: https://www.softwaretestinghelp.com/machine-learning-tools/.

[2] A. Al-Masri. *What Are Training, Validation and Test Data Sets in Machine Learning?* 2018. URL: https://medium.com/datadriveninvestor/what-are-training-validation-and-test-data-sets-in-machine-learning-d1dd1ab09bae.

[3] Amazon. URL: https://www.amazon.com/gp/help/customer/display.html?nodeId=GE4KRSZ4KAZZB4BV.

[4] R. Baker. *Starbucks on first name terms*. 2012. URL: https://www.marketingweek.com/starbucks-on-first-name-terms/.

[5] *Basic Concepts in Machine Learning*. 2020. URL: https://machinelearningmastery.com/basic-concepts-in-machine-learning/.

[6] P. Cooper. *How Does the YouTube Algorithm Work? A Guide to Getting More Views*. 2019. URL: https://blog.hootsuite.com/how-the-youtube-algorithm-works/.

[7] *Developer Guide*. URL: https://developers.google.com/maps/documentation/geocoding/intro.

[8] M. Grčar, D. Mladenič e M. Grobelnik. *User Profiling for the Web*. 2006. URL: http://elib.mi.sanu.ac.rs/files/journals/csis/6/030201.pdf.

[9] *JqueryEvent*. URL: https://api.jquery.com/category/events/event-object/.

[10] S. Kanoje, S. Girase e D. Mukhopadhyay. "User Profiling Trends, Techniques and Applications". Em: *CoRR* abs/1503.07474 (2015). arXiv: 1503.07474. URL: http://arxiv.org/abs/1503.07474.

[11] J. Nielsen. *A 100-Year View of User Experience*. 2017. URL: https://www.nngroup.com/articles/100-years-ux/.

[12] V. Singh. *HTTP Request*. 2017. URL: https://www.toolsqa.com/client-server/http-request/.

[13] *Understanding Cookies and Sessions*. URL: http://www.lassosoft.com/Tutorial-Understanding-Cookies-and-Sessions.

[14] *Website Personalization*. URL: https://www.optimizely.com/optimization-glossary/website-personalization/.

[15] Zyxo. *Why decision trees is the best data mining algorithm*. 2010. URL: https://zyxo.wordpress.com/2010/09/17/why-decision-trees-is-the-best-data-mining-algorithm/.