INSTITUTO POLITÉCNICO
DE VIANA DO CASTELO

Rui José da Rocha Lima

EXTRAÇÃO E ANÁLISE MULTIDIMENSIONAL DE DADOS DE ATLETISMO A PARTIR DE DADOS NÃO ESTRUTURADOS

Mestrado de Engenharia de Software

Trabalho de projeto efetuado sob a orientação de:
Doutora Maria Estrela Ribeiro Ferreira da Cruz

Fevereiro de 2018

RESUMO

Este projeto propõe o design e implementação de um armazém de dados (DW) composto por resultados de atletismo a nível distrital e nacional, a partir de tabelas criadas inicialmente para interpretação humana. O projeto propõe também um mecanismo de integração entre os dados sobre as marcas obtidas em provas de atletismo com a sua localização geográfica (altitude incluída) e as condições atmosféricas em que as provas foram realizadas. Os ficheiros de onde os dados sobre os resultados das provas são originários encontram-se em múltiplos formatos (.xls, .txt, .doc) mas o predominante é ".pdf". Os ficheiros estão distribuídos por 20 distritos mais os resultados ao nível nacional.

Este trabalho de investigação está dividido em duas grandes partes. Uma parte envolve o desenvolvimento do protótipo de um módulo *Python*, chamado de *PositionParser*, que servirá de *Framework* para a identificação e extração dos valores dos campos do texto. O processo de análise do texto envolve a sua "tokenização", marcação e visualização dos dados "tokenizados", segmentação e definição da ordem de leitura, extração e definição da hierarquia dos dados.

Outra parte envolve todo o processo de análise de dados: o *Web scraping* dos documentos com os resultados das competições, a conversão de todos os documentos para o formato PDF e posteriormente para ficheiros de texto, a extração de dados das tabelas não formatadas dos ficheiros de texto recorrendo ao protótipo criado e a limpeza, uniformização e armazenamento dos dados num DW.

Fevereiro de 2018

ABSTRACT

This project proposes the design and implementation of a data warehouse (DW) composed by athletic results at a national and district level, from data tables initially crated for human interpretation. The project also proposes an integrating mechanism between athletic records with their geographic location (altitude included) and atmospheric conditions of the competitions.
The original files, about competition results, come in multiple formats (.xls, .txt, .doc) but the predominant one is ".pdf". The files are spread across the 20 Portuguese districts plus results at a national level.
This research work is divided in two main parts. One part involves developing a Python prototype module, called *PositionParser* that can serve has a framework for identification and extraction of field values from text. The parsing process involves tokenizing the text, visualizing tokenized data, segmentation and definition of the reading order, extracting data and definition of the data hierarchy.
The other part involves the whole data analysis process: web scraping of the documents with the competition results, conversion of all documents into PDF format and subsequently into plain text files, extraction of the data from the unformatted tables in the resulting text files with the help of the prototype, cleansing, standardization and storage of the information into a DW.

Fevereiro de 2018

Agradecimentos

A toda a minha família, incluindo os que já partiram, pelo seu apoio.

À doutora Maria Cruz pela sua orientação.

CONTEÚDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE ACRÓNIMOS

DW – Data Warehouse
PDF – Portable Document File
CSV – Comma Separated File
JSON – JavaScript Object Notation
DSR – Design Science Research
BI – Business Intelligence
DM – Data Mining
API – Application Programming Interfaces
ETL – Extract, Transform and Load
URL – Uniform Resource Locator
XML – Extensible Markup Language
XPath – XML Path
MD5 – Message Digest 5
GREL – General Refine Expression Language
HTML – HyperText Markup Language

1. INTRODUÇÃO

Hoje em dia muitas associações, empresas e instituições colocam grande parte da sua informação online. Assim, Informação relacionada e com o mesmo tipo de dados encontra-se distribuída por múltiplos *Web sites* e disponível para qualquer pessoa. Entre esses casos estão os resultados das provas de atletismo disponibilizados pelas várias associações distritais portuguesas. Agrupar toda esta informação num Data Warehouse (DW) seria de grande utilidade para uma análise posterior, no entanto essa informação encontra-se publicada em ficheiros do tipo *portable document format* (PDF). O formato PDF e a ausência de uma estrutura aparente da informação impede que a informação seja processada automaticamente, logo é necessário um processamento inicial antes que possa ser transferida para um DW.

1.1 CONTEXTO E MOTIVAÇÃO

A área de desporto possui muitas associações que disponibilizam um volume relativamente grande de informação aos seus atletas e sobre as provas realizadas por estes ao público em geral. Cada associação desportiva (distritais) disponibiliza informação sobre as provas organizadas e realizadas por estas, no entanto, a estruturas dos dados usada para apresentar essa informação nem sempre é a mesma. Agrupar esses dados e interliga-los entre si para efetuar uma análise posterior seria interessante.

Dentro da área do desporto, o ramo do atletismo foi escolhido para esta investigação porque, para além de ser um dos desportos com mais participantes federados em Portugal (http://www.pordata.pt/Portugal/Praticantes+desportivos+federados+total+e+por+algumas+federa%C3%A7%C3%B5es+desportivas-2226. Acedido em 23 de Novembro de 2017), é uma prática individual, o que permite obter múltiplos resultados a partir de uma única prova e seguir com algum grau de confiança (exceto em casos de provas combinadas e passagens de testemunho) a evolução de um atleta numa modalidade, ao contrário de outros desportos mais populares de equipa, como o futebol, basquetebol ou andebol (ver gráfico da figura 1). Por ser um desporto "base" os resultados apresentados estão mais perto do rendimento real quando comparados com fatores ambientais (como humidade ou temperatura), ao contrário de outros desportos (como por exemplo o ténis) onde fatores como estratégia, arbitragem e adversário têm um peso maior.

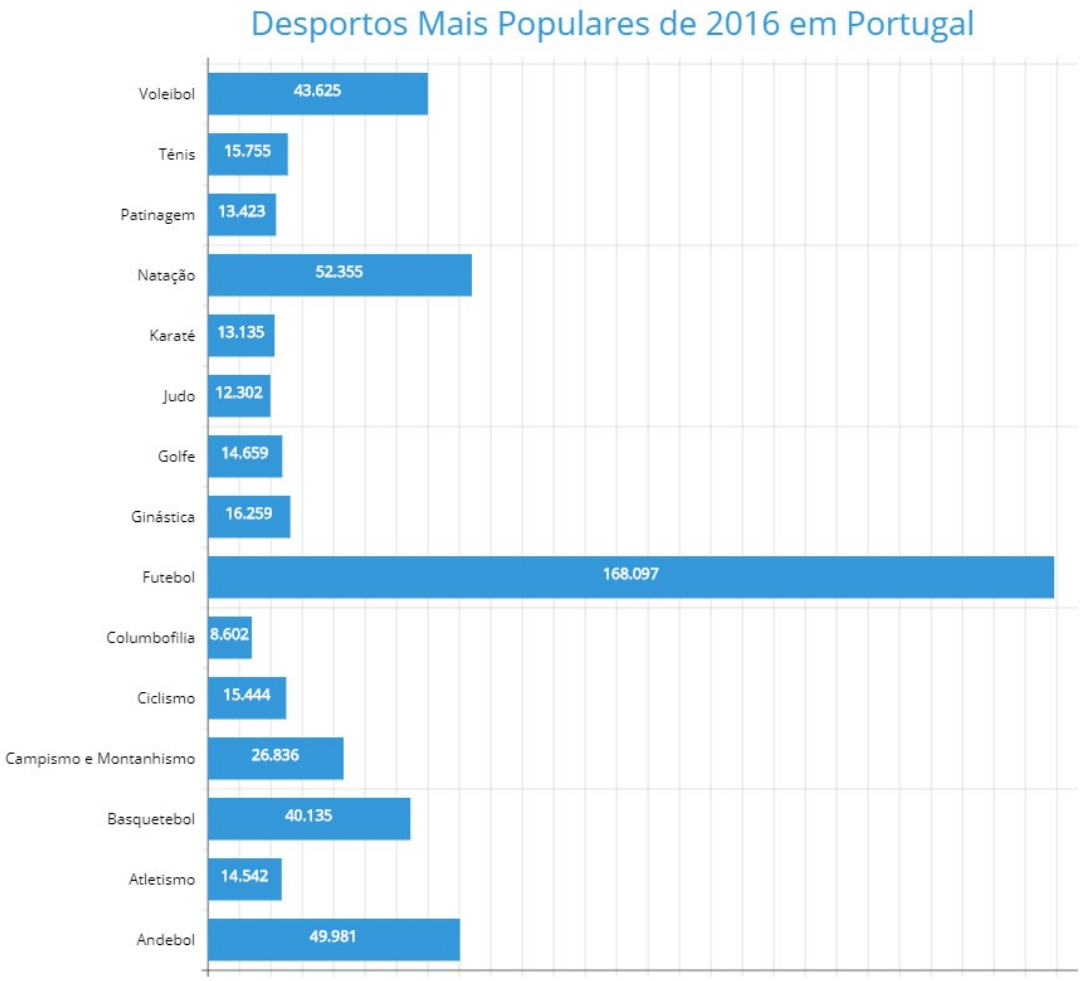


Figura 1 – Quantidade de Praticantes dos Desportos Atualmente mais Populares

Analisar resultados do atletismo português fornece uma oportunidade única de criar valor, pois como as associações desportivas são, na maioria das vezes, não lucrativas (http://www.aaaveiro.pt/docs/estatutos.0001.docs.pdf. Acedido em 12 de Julho de 2017) (http://adac.pt/wp-content/uploads/2017/03/Estatutos-ADAC.pdf. Acedido em 12 de Julho de 2017), têm pouco incentivo, ou verba, para melhorar os seus serviços. Assim, a informação disponível ao público encontra-se na maior parte das vezes desestruturada e em formato PDF, sendo por isso, de tratamento difícil e trabalhoso. No entanto, criar um DW centralizado com resultados nacionais e distritais poderia cativar partes interessadas como equipas de atletismo, treinadores, atletas, entusiastas do desporto ou até mesmo de investigação científica. Esse é o principal objetivo desta investigação.

1.2 Objetivos de Pesquisa e Plano de Trabalho

O principal objetivo deste trabalho é agregar num DW um conjunto de dados recolhidos de diversas associações desportivas sobre as provas de atletismo realizadas, pontuações e atletas envolvidos, como é possível observar na figura 2.

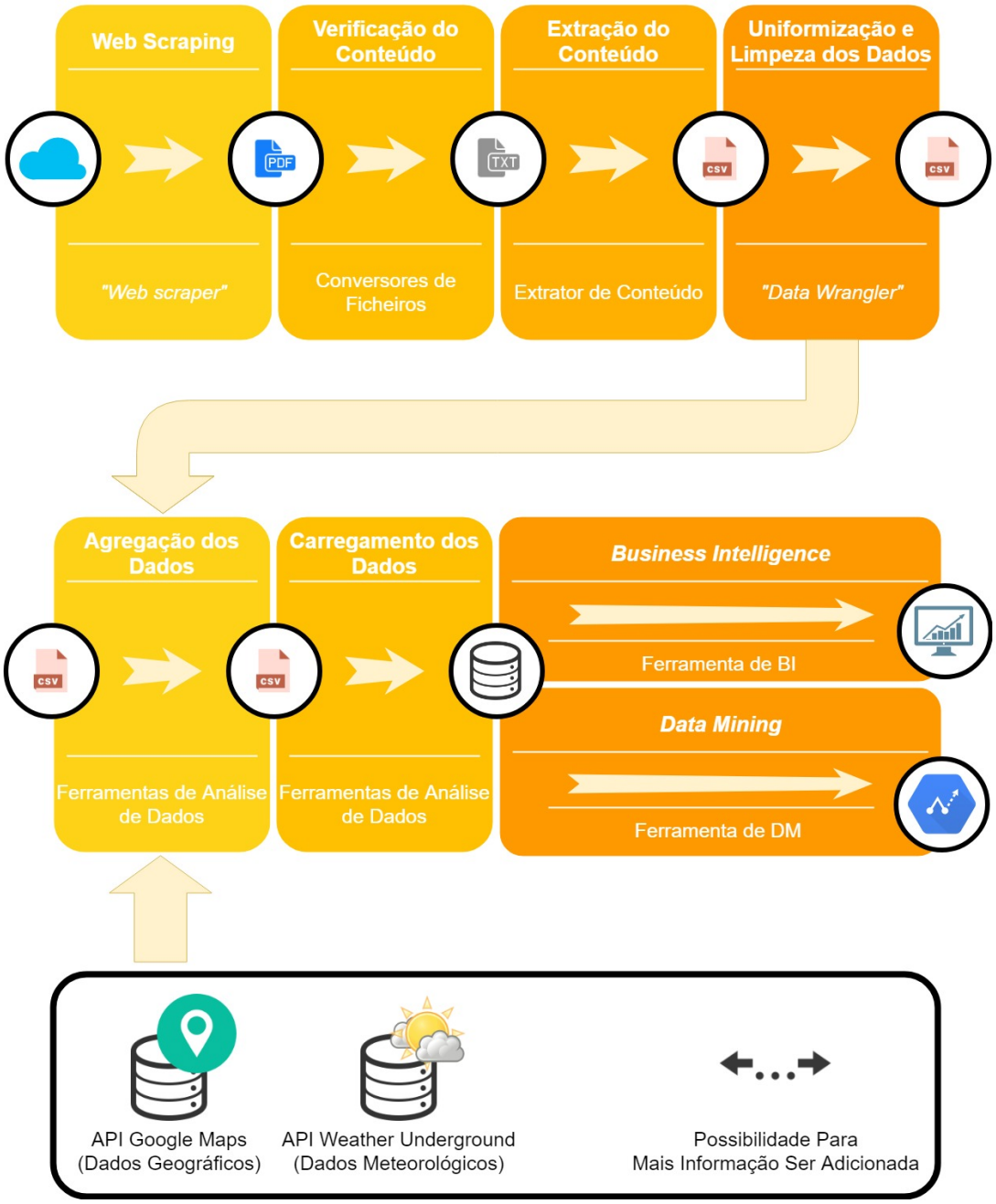


Figura 2 – Esquema Geral do Projeto

O processo de análise dos resultados de atletismo a partir dos Web sites das respetivas associações envolve:

1. O *web scraping* dos ficheiros, encontrando os seus *links* a partir de todas as associações de atletismo, verificando se os ficheiros já foram anteriormente carregados e guardando os novos ficheiros;

2. A conversão dos ficheiros para o formato de texto simples, recorrendo a uma ferramenta para converter ficheiros que não sejam PDF para PDF e estes para texto plano;
3. A utilização do *PositionParser* para extrair dados num formato organizado, criando scripts para analisar as tabelas em texto simples;
4. A limpeza dos dados, removendo duplicados, definindo os tipos de colunas e uniformizando os dados;
5. O aumento dos dados, cruzando informação geográfica da API *Google Maps* e informação meteorológica da API *weather underground*;
6. O carregamento dos dados para um DW;
7. Possível análise dos dados recorrendo a técnicas de *Business Intelligence* (BI) e *Data Mining* (DM).

No ponto 3 a ferramenta foi criada por mim, uma vez que não foi encontrada uma solução satisfatória, demonstrado no capítulo 3.3 da análise do estado da arte.

1.3 Metodologia de Investigação

O método de investigação usado neste trabalho é o *Design Science Research* (DSR). A metodologia DSR é frequentemente aplicada no desenvolvimento e design de artefactos para resolver problemas de organizações, possibilitando, direta ou indiretamente, o aumento dos seus lucros (Hevnert et al., 2004). Logo, um projeto de investigação que use DSR precisa da criação intencional de um artefacto inovador para resolver um problema específico num determinado domínio (Hevneret et al., 2004).

O DSR recorre a uma abordagem iterativa composta por seis atividades principais (Pefferset et al., 2006):

1. Identificação do problema e motivação, definido no capítulo 1.1;
2. Identificação dos objetivos e solução, que foram especificados no capítulo 1.2;
3. Design e desenvolvimento da solução, que será apresentado no capítulo 4;
4. Demonstração, onde a utilização do protótipo criado (*PositionParser*) demonstra que é possível obter os resultados desportivos de uma grande quantidade de ficheiros;
5. Avaliação, onde são examinados os resultados do DW produzido;
6. Comunicação, onde se pretende escrever um artigo em que se pretende apresentar os resultados desta investigação. Pretende-se que o artigo seja publicado numa conferência internacional da área.

O DSR é uma abordagem iterativa e incremental, permitindo que o utilizador aperfeiçoe o artefacto criado através de feedback e construção. DSR pode começar como uma representação simplificada do problema e, com a evolução do ambiente e tecnologia disponível, hipóteses anteriores podem ser comprovadas ou podem tornar-se inválidas (Von Alan et al., 2004).

1.4 Estrutura do Documento

Este trabalho de dissertação está organizado num total de seis capítulos:

Capítulo 1: é o capítulo corrente onde foi feita uma contextualização do trabalho a ser desenvolvido, foram apresentados os objetivos e motivação da investigação e foi apresentada a metodologia de investigação que vai ser usada neste trabalho.

Capítulo 2: neste capítulo apresentam-se alguns conceitos e definições usados neste projeto e faz-se uma comparação entre eles;

Capítulo 3: neste capítulo faz-se uma comparação entre ferramentas, tecnologias semelhantes ao protótipo de análise de dados desenvolvido (*PositionParser*) e entre as bases de dados centralizadas de atletismo já existentes;

Capítulo 4: neste capítulo dá-se uma explicação do processo de análises de dados usado para criar o DW e apresenta-se o protótipo *PositionParser* que irá analisar e relacionar o conteúdo dos ficheiros recebidos e gerar ficheiros no formato CSV;

Capítulo 5: neste capítulo apresentam-se os resultados da análise aos dados do DW criado e carregado;

Capítulo 6: neste capítulo apresentam-se as conclusões do trabalho e apresentam-se algumas linhas para trabalho futuro.

2. Definições e Conceitos

Neste capítulo serão apresentados alguns conceitos e definições que serão usados durante o trabalho de investigação aqui apresentado.

2.1 Data Wrangling

Data *wrangling* é um processo que permite transformar um conjunto de dados desorganizados e confusos, não refinados, em dados acessíveis e úteis. Os seus processos envolvem a extração de dados no seu estado natural (muitas vezes através de APIs ou *Web scraping*), análise do conteúdo extraído e uniformização e limpeza dos dados. Este processo permite ter acesso a informação que mais ninguém se incomodaria a procurar e torná-la clara e implementável (Kazil, J. & Jarmul, K., 2016).
Este é o processo, usado neste trabalho, que torna os dados sobre os resultados desportivos utilizáveis para análise.

2.2 Web Scraping

Web scraping é o termo usado quando um software é utilizado para extrair dados de *websites*. A filosofia do *Web scraping* consiste em recolher informação recorrendo a qualquer meio que não seja um programa a interagir com uma API ou um ser humano a interagir com um *Web browser*. Isto costuma ser atingido escrevendo programas automáticos que sondam um servidor *Web*, solicitam conteúdo e decompõem-no para extrair os elementos desejados (Mitchell, R., 2015).
No âmbito do trabalho apresentado aqui, todos os resultados desportivos tratados são originários de documentos obtidos através deste processo.

2.3 Limpeza dos Dados

A limpeza de dados é um processo que envolve a sua normalização ao longo das colunas ou apenas de alguns valores, o cálculo de novos valores a partir dos valores atuais ou até mesmo definir o tipo de dados de cada coluna (Kazil, J. & Jarmul, K., 2016).
A uniformização ou normalização dos dados (não confundir com a normalização duma base de dados relacional ou normalização estatística) é o processo que garante que valores linguisticamente ou logicamente equivalentes entre si são apresentados, ou pelo menos comparados, equivalentemente (Mitchell, R., 2015). Por exemplo os valores “(258) 123 456” e “258-123456”, num determinado contexto, representam o mesmo número de telefone, logo deverão ser normalizados.
Os dados extraídos são limpos antes de poderem ser analisados e/ou armazenados num DW.

2.4 Data Warehouse

Um DW é um repositório utilizado para a consolidação da informação num formato válido e consistente, permitindo que os utilizadores analisem os dados de forma seletiva através de técnicas de BI para exploração do DW e DM que integra conceitos da área de estatística e inteligência artificial (Santos, M. Y. & Ramos, I., 2009).A informação nele contida é não volátil, orientada apenas a um assunto e catalogada cronologicamente (Santos, M. Y. & Ramos, I., 2009).

Um DW é um repositório central que serve de entrada aos mecanismos de tomada de decisão e que, por norma, contém uma grande quantidade de dados de múltiplas origens (Caldeira, C. P., 2012).

No trabalho aqui apresentado, vai ser conceptualizado e criado um DW que servirá como repositório dos dados recolhidos e analisados.

2.5 ETL

ETL (Extract Transform Load) é um processo constituído por três passos principais: Extração (Extract) dos dados dos sistemas fonte; Transformação (Transform) do formato dos dados para o formato dos dados no DW; Carregamento (Load) dos dados no DW.

No trabalho aqui documentado, após a sua preparação, os dados são transferidos através deste processo, para um DW modelado num esquema em estrela. O esquema em estrela é representado por um diagrama de modelo de dados dimensional composto por uma tabela central de factos ou medidas e relacionada com várias tabelas de dimensões (Caldeira, C. P., 2012).

2.6 Business Intelligence

Business Intelligence (BI) é um processo que permite a tomada de decisões de negócio "inteligentes" baseadas na análise dos dados disponíveis.

Os sistemas de BI combinam ferramentas analíticas para fornecer informação importante na tomada de decisão através da exploração de uma grande quantidade de dados armazenados numa base de dados organizacional (Santos, M. Y. & Ramos, I., 2009).

Este processo é utilizado no DW concebido no projeto aqui apresentado.

2.7 Data Mining

DM (*Data Mining*) é um processo que procura identificar padrões relevantes numa determinada forma de representação, como por exemplo árvores de decisão, regressão, segmentação e assim por diante (Fayyad et al., 1996).

A grande diferença entre DM e outras ferramentas de análise de dados de BI está na forma como os dados são explorados. Enquanto em BI o utilizador constrói uma

hipótese e confirma-a, ou refuta-a, recorrendo a uma análise da relação entre os dados. No processo DM, este fica encarregue da geração das próprias hipóteses, o que garante uma maior rapidez, autonomia e fiabilidade da análise, pois o modelo não fica dependente da intuição e habilidade do analista (Santos, M. F. & Azevedo, C., 2005).

2.8 DOCUMENTO PDF

PDF (*Portable Document Format*) é um formato de ficheiro que capta os elementos de um documento impresso como uma imagem eletrónica. Este formato de ficheiro é um dos mais usados na publicação de documentos online. Os dados armazenados em tabelas dentro de ficheiros no formato PDF são, na maioria das vezes, difíceis de analisar automaticamente, ou de ser interpretados por maquinas.

Normalmente são evitadas fontes de dados em formatos de difícil análise, no entanto, se não for possível evitar lidar com ficheiros PDF, será necessário aprender a analisá-los, o que poderá ser um processo notoriamente difícil (Kazil, J. & Jarmul, K., 2016).

Ficheiros neste formato podem ser facilmente interpretados por humanos que podem fazer deduções sobre os dados armazenados, mas de difícil análise por software. No entanto na grande maioria das vezes este formato de dados é o único disponível. Algumas das razões apontadas para isso são:

1. O dono do documento poderá não querer que o público reutilize os seus dados;
2. É esperado que os dados sejam apenas utilizados por humanos;
3. O dono dos dados poderá querer que estes não sejam facilmente editados pelo leitor;
4. O dono apresenta os dados apenas neste formato porque pode ser lido pela maioria dos *Web browsers* modernos sem recorrer a aplicações externas;
5. O dono apresenta os dados apenas neste formato por ser fácil de utilizar e criar;
6. O dono apresenta os dados apenas neste formato por estar amplamente disponível.

Existem ferramentas de software capazes de extrair o conteúdo textual de documentos PDF, no entanto é improvável que o resultado reflita corretamente a estrutura original da tabela (McCallum, Q., 2013).

Na era digital, muita da informação é partilhada online. O formato mais comum utilizado na partilha de dados é o PDF (Khusro, S., Latif, A., & Ullah, I., 2015). Documentos PDF são portáteis, o que significa que, são independentes da plataforma e da aplicação utilizada na sua produção (Bienz, T., Cohn, R., & Adobe Systems, 1993). O formato PDF foi pensado para ser interpretado por humanos, e não está preparado para ser operado automaticamente por aplicações de software. Consequentemente, deteção de tabelas e extração de dados de ficheiros PDF não é uma tarefa simples, mas cada vez mais requisitada. Várias abordagens foram propostas para detetar e extrair tabelas disponíveis em ficheiros PDF, como é o caso de (Khusro, S., Latif, A., &

Ullah, I., 2015), (Yildiz, B., Kaiser, K., & Miksch, S., 2005), (Oro, E., & Ruffolo, M., 2009), (Pitale, S., & Sharma, T., 2011) e (Hassan, T., & Baumgartner, R., 2007).
A grande maioria de ficheiros onde se encontram os dados extraídos para análise neste trabalho, encontra-se no formato PDF.

2.9 JSON

JSON (*JavaScript Object Notation)* é uma linguagem de marcação baseada em texto criada tanto para leitura humana como para análise computacional (McCallum, Q., 2013).
Recentemente JSON tornou-se mais popular do que, por exemplo, XML por apresentar a mesma informação num formato mais pequeno (Mitchell, R., 2015). JSON é um dos formatos mais usados para transferência de dados (Kazil, J. & Jarmul, K., 2016). Este standard aberto também é usado na transferência de dados de *Application Programming Interfaces* (APIs). Este é o padrão utilizado pelas APIs de dados meteorológicos e geográficos de onde serão transferidos dados.

2.10 CSV

CSV (*comma-separated values*) é um formato de ficheiros onde se armazenam dados de texto e numéricos simples. Neste tipo de ficheiros cada linha corresponde a um registo, sendo cada registo constituído por vários campos separados por uma vírgula. Ficheiros CSV são muito comuns em Web scraping (Mitchell, R., 2015) e provavelmente os mais utilizados para armazenar dados acabados de extrair. Para que este formato seja respeitado, cada registo tem a sua própria linha, cada campo está separado por uma vírgula e alguns ou todos os valores poderão estar entre cotações ou outros caracteres para o proteger (McCallum, Q., 2013).

2.11 PYTHON

Python é uma linguagem de programação de alto nível e orientada a objetos. A influência desta linguagem está a aumentar tanto no meio académico como nas empresas. Esta linguagem é usada por um número significativo de universidades (incluindo portuguesas) como forma de introduzir ciências de computação a futuros engenheiros de software e programadores. Ao nível das empresas, *Python* tem sido principalmente usado para desenvolver jogos de vídeo, aplicações *Web* e de análise de dados (Vasconcelos, J., 2015).
No trabalho apresentado neste documento, *Python* é a linguagem utilizada na análise, limpeza e carregamento dos dados.

3. Análise do Estado da Arte

Neste capítulo serão focadas abordagens e ferramentas diferentes para cada fase do projeto. No subcapítulo 3.1 são comparadas ferramentas de *Web scraping*, no 3.2 são comparados conversores de ficheiros, no 3.3 trata de ferramentas de extração de dados, no 3.4 compara ferramentas de limpeza de dados, no 3.5 faz-se um levantamento de ferramentas de análise de dados. No subcapítulo 3.6 faz-se um levantamento de APIs externas para fontes de dados e no subcapítulo 3.7 de ferramentas de BI e DM.

3.1 Web Scraping

Para extrair os ficheiros com os resultados de atletismo é necessário escolher qual a melhor ferramenta de *Web scraping* para o trabalho. As várias ferramentas vão ser analisadas e comparadas sobre os seguintes aspetos: preço; tipo de licença; se tem, ou não, uma interface gráfica que facilite a sua utilização; se possui ou não uma API capaz de tornar programável as ações da aplicação e se efetua buscas recursivas de páginas Web (*Web crawling*).

	Import.io	**Webhose.io**	**Dexi.io**	**Scrapy**	**Selenium Webdriver**	**Ghost.py**
Web Crawling (Web spider)	Sim	Sim	Sim	Sim	Não	Não
API	Web API	Web API	Web API	Sim	Sim	Sim
Interface Gráfica	Sim	Sim	Sim	Não	Não	Não
Preço	De 169$ a 419$ por mês	Gratuito para apenas 1000 pedidos por mês, podendo chegar a 4000$ ou mais	De 105$ a 699$ por mês	Gratuito	Gratuito	Gratuito
Licença	Proprietária	Proprietária	Proprietária	BSD-3	Apache-2	MIT

Tabela 1 – Comparação de Ferramentas e Serviços de Web Scraping

Como é possível observar na tabela 1, as ferramentas "import.io", "Webhose.io" e "Dexi.io" apenas disponibilizam a sua solução como um serviço e não como algo permanente. Assim, se as quisermos integrar programaticamente é necessário recorrer à sua *Web API.* Além disso, temos que acrescentar o facto de ser necessário um pagamento mensal ou por pedidos. Pelas razões apontadas, estas ferramentas não foram selecionadas. Por outro lado, "Scrapy", "Selenium Webdriver" e "Ghost.py" possuem licenças menos restritivas e as suas APIs podem ser acedidas programaticamente com *Python* sem recorrer a pedidos Web.

Como *Web Scraper* principal foi escolhido o "Scrapy" por ser considerado o mais poderoso, por ser escrito em *Python*, ter um motor assíncrono, ser extremamente rápido e capaz de lidar com grandes quantidades de tarefas (Kazil, J. & Jarmul, K., 2016). Neste projeto é necessário pesquisar múltiplas páginas de cada *Web site* até se encontrar os ficheiros desejados, e das soluções consideradas, o "Scrapy" é a única a possibilitar esse tipo de análise recursiva (*Web crawling)* (Kazil, J. & Jarmul, K., 2016) sem apresentar custos monetários. É relativamente fácil de usar, no entanto, necessita sempre de uma configuração inicial para cada "*crawler"* (Mitchell, R., 2015).

Existem casos em que um *Web site* utiliza uma grande quantidade de JavaScript ou outro tipo de código que preencha uma página *Web* com conteúdo, após esta já ter sido carregada (Kazil, J. & Jarmul, K., 2016). Nesses casos *scrapers* convencionais não são capazes de realizar a análise, sendo necessário simular um *browser* que leia o ecrã e não o conteúdo de uma página Web como por exemplo o "Selenium Webdriver" e o "Ghost.py" (Kazil, J. & Jarmul, K., 2016). Entre estes dois, o "Selenium" foi escolhido porque, apesar de ter sido inicialmente criado para testes de *Web sites* (Mitchell, R., 2015) é a ferramenta mais popular (Kazil, J. & Jarmul, K., 2016). Isto geralmente significa uma maior comunidade para a resolução de problemas de utilização. O "Ghost.py",para além de não ser tão popular, necessita de uma grande quantidade de bibliotecas (Kazil, J. & Jarmul, K., 2016), o que geralmente significa perder muito tempo em torno de dependências.

Para esta fase, que se foca em Web scraping, as ferramentas selecionadas foram as seguintes:

1. Scrapy – para *Web scraping/crawling*;
2. Selenium Webdriver – nos casos em que a informação desejada é carregada dinamicamente.

3.2 Conversão do Tipo de Ficheiro

Antes do conteúdo poder ser analisado é necessário que os ficheiros tenham um mesmo formato, de preferência um formato fácil de analisar automaticamente, por exemplo o formato .txt. Para isso, é necessário recorrer a conversores de um tipo de ficheiros para outro.

Na tabela 2 faz-se uma análise e comparação entre ferramentas que convertem ficheiros de/para diferentes formatos. Uma vez que o formato predominante dos ficheiros origem é PDF, é dada uma maior ênfase aos conversores que trabalham com esse tipo de ficheiros. As ferramentas vão ser analisadas e comparadas nos seguintes aspetos: tipo de ficheiros tratados (ficheiros origem e ficheiros destino), se mantém ou não o *layout*, preço e tipo de licença.

	LibreOffice	**pdftotext**	**pdf2txt (PDFMiner)**	**PDF to Text (web)**
Tipo de Ficheiros de Origem	Múltiplos Formatos	PDF	PDF	PDF
Tipo de Ficheiros de Destino	Múltiplos Formatos	TXT	TXT	TXT, DOC, DOCX
Mantém o Layout	Não	Sim	Não	Não
Preço	Gratuito	Gratuito	Gratuito	Gratuito
Licença	MPL-2	GPL-2/3	MIT	Proprietária

Tabela 2 – Comparação de ferramentas de Conversão de Ficheiros

Como se pode ver na tabela 2, foram analisados os seguintes conversores:

- "PDF to Text" é um conversor de PDF para ficheiros de texto, disponível *on line* (http://pdftotext.com. Acedido em 15 de Janeiro de 2018), capaz de converter até 20 ficheiros de uma vez.
- A solução "pdf2txt" é um comando disponível após a instalação do "PDFMiner" que converte ficheiros PDF em ficheiros de texto simples (Kazil, J. & Jarmul, K., 2016).
- LibreOffice é um conjunto de aplicações, de código aberto, compatíveis com uma grande quantidade de formatos de documentos, incluindo originários do Microsoft Office (LIBREOFFICE, 2017).
- Outra solução capaz de converter ficheiros PDF em texto é o "pdftotext", com a vantagem de ser também possível tentar manter o *layout* original do texto, um fator indispensável para a análise do passo seguinte.

Apesar de na conversão os dados se manterem intactos, é virtualmente impossível recuperar de forma confiável a correspondência entre os valores dos dados assim como o seu significado (McCallum, Q., 2013). Este problema vai tentar ser ultrapassado recorrendo à análise da fase 3.

A solução encontrada consiste na utilização das ferramentas seguintes:

1. LibreOffice – Através da linha de comandos converter, programaticamente, todos os ficheiros que não sejam PDF ou TXT para PDF;
2. pdftotext – Através da linha de comandos converter, programaticamente, todos os ficheiros PDF para TXT.

3.3 Extração do Conteúdo

Na tabela 3, faz-se uma análise e comparação entre ferramentas existentes para extrair dados de tabelas armazenadas em ficheiros no formato PDF.

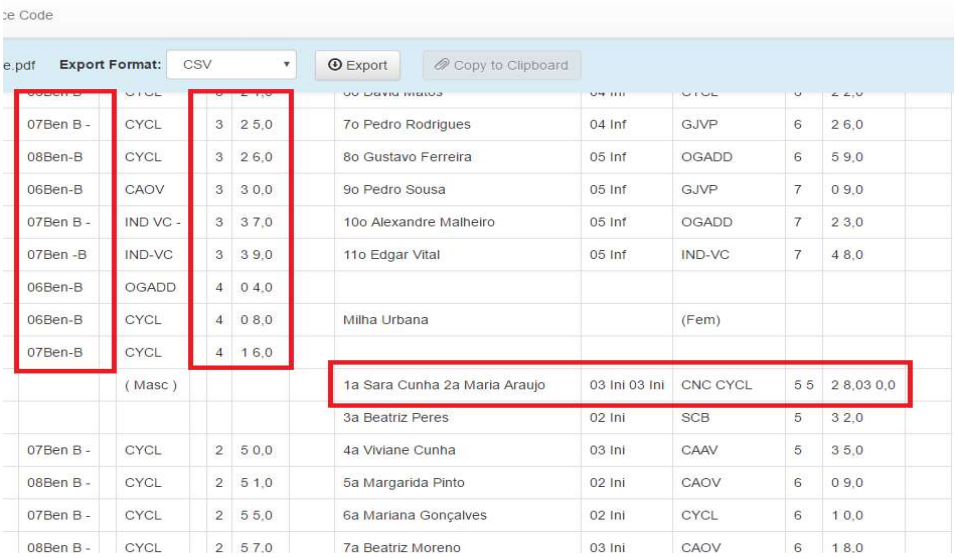
	Tabula	**PDFTables**	**PDFix**
Identificação dos Campos	Não	Não	Não
Ordem de Leitura e Deteção de Tabelas	Manual Auto	Auto	Auto
Hierarquia dos Campos	Não	Não	Auto
Extração para um Formato Estruturado	Sim	Sim	Sim
Escalabilidade (API)	Não	Sim	Sim
Ficheiros de Origem	PDF	PDF	PDF
Interface Gráfica	Sim	Sim	Não
Preço	Gratuito	De 0.02$ a 0.03$ cada página	De 490$ a 2270$

Tabela 3 – Comparação de Ferramentas de Extração de Conteúdo

"Tabula" é uma ferramenta gratuita e de código aberto para extrair dados. Esta ferramenta foi criada por jornalistas para dar a possibilidade aos jornalistas de trabalhar com informação "trancada" em PDFs (http://tabula.technology. Acedido em 13 de Julho de 2017). Esta ferramenta possui uma seleção automática onde o mesmo algoritmo é usado para seccionar as tabelas, como é possível observar na figura 3. No entanto, recorrendo à seleção manual não é possível escalar o trabalho. Além disso, também não permite definir as hierarquias dos campos ou as relações entre eles, nem definir a ordem de leitura do texto como um todo.

Figura 3 – Tabelas Mal Identificadas por Tabula

A análise é feita automaticamente o que, muitas vezes, não facilita uma segmentação apropriada. Nos testes efetuados no âmbito da investigação foi comum observar situações onde 2 campos eram colocados na mesma coluna, 1 campo separado em 2 colunas ou 2 entradas colocadas na mesma linha, como reproduzido na figura 4.

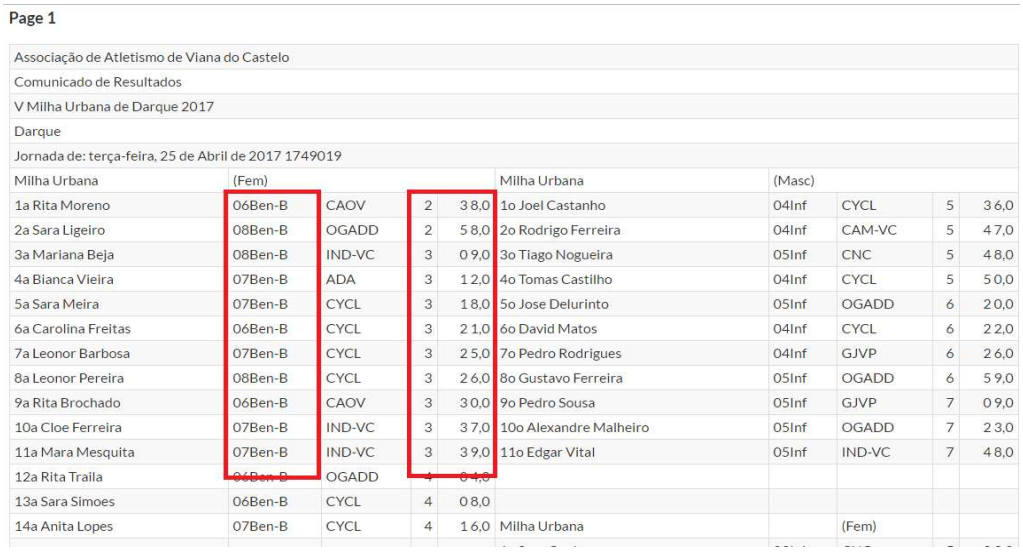


07Ben B -	CYCL	3	2 5,0	7o Pedro Rodrigues	04 Inf	GJVP	6	2 6,0
08Ben-B	CYCL	3	2 6,0	8o Gustavo Ferreira	05 Inf	OGADD	6	5 9,0
06Ben-B	CAOV	3	3 0,0	9o Pedro Sousa	05 Inf	GJVP	7	0 9,0
07Ben B -	IND VC -	3	3 7,0	10o Alexandre Malheiro	05 Inf	OGADD	7	2 3,0
07Ben -B	IND-VC	3	3 9,0	11o Edgar Vital	05 Inf	IND-VC	7	4 8,0
06Ben-B	OGADD	4	0 4,0					
06Ben-B	CYCL	4	0 8,0	Milha Urbana		(Fem)		
07Ben-B	CYCL	4	1 6,0					
	(Masc)			1a Sara Cunha 2a Maria Araujo	03 Ini 03 Ini	CNC CYCL	5 5	2 8,03 0,0
				3a Beatriz Peres	02 Ini	SCB	5	3 2,0
07Ben B -	CYCL	2	5 0,0	4a Viviane Cunha	03 Ini	CAAV	5	3 5,0
08Ben B -	CYCL	2	5 1,0	5a Margarida Pinto	02 Ini	CAOV	6	0 9,0
07Ben B -	CYCL	2	5 5,0	6a Mariana Gonçalves	02 Ini	CYCL	6	1 0,0
08Ben B -	CYCL	2	5 7,0	7a Beatriz Moreno	03 Ini	CAOV	6	1 8,0

Figura 4 – Dados Mal Analisados por Tabula

Mesmo que a seleção e análise seja feita corretamente, o seu maior problema mantém-se, que é o fato do seu processo ser feito manualmente, ficheiro a ficheiro, através da interface gráfica. Desta forma, não permite que o utilizador final escale o processo. Apesar de possuir uma interface simples, esta ferramenta nem sempre é uma boa solução porque o ficheiro CSV gerado costuma apresentar uma estrutura desorganizada ou confusa (Kazil, J. & Jarmul, K., 2016).

O “PDFTables” deixou de ser suportado pelos criadores originais e é atualmente disponibilizado como um serviço (Kazil, J. & Jarmul, K., 2016) que converte ficheiros PDF em XLSX. Este serviço pode ser acedido através duma API, permitindo assim escalar o trabalho (https://pdftables.com. Acedido em 13 de Julho de 2017). Esta ferramenta apenas faz a análise e seleção de tabelas automaticamente. Em alguns dos testes, efetuados no âmbito deste trabalho, foram detetados 2 campos na mesma coluna e 1 campo em 2 colunas (como se pode ver na figura 5).



Page 1

Associação de Atletismo de Viana do Castelo
Comunicado de Resultados
V Milha Urbana de Darque 2017
Darque
Jornada de: terça-feira, 25 de Abril de 2017 1749019

Milha Urbana	(Fem)				Milha Urbana	(Masc)			
1a Rita Moreno	06Ben-B	CAOV	2	38,0	1o Joel Castanho	04Inf	CYCL	5	36,0
2a Sara Ligeiro	08Ben-B	OGADD	2	58,0	2o Rodrigo Ferreira	04Inf	CAM-VC	5	47,0
3a Mariana Beja	08Ben-B	IND-VC	3	09,0	3o Tiago Nogueira	05Inf	CNC	5	48,0
4a Bianca Vieira	07Ben-B	ADA	3	12,0	4o Tomas Castilho	04Inf	CYCL	5	50,0
5a Sara Meira	07Ben-B	CYCL	3	18,0	5o Jose Delurinto	05Inf	OGADD	6	20,0
6a Carolina Freitas	06Ben-B	CYCL	3	21,0	6o David Matos	04Inf	CYCL	6	22,0
7a Leonor Barbosa	07Ben-B	CYCL	3	25,0	7o Pedro Rodrigues	04Inf	GJVP	6	26,0
8a Leonor Pereira	08Ben-B	CYCL	3	26,0	8o Gustavo Ferreira	05Inf	OGADD	6	59,0
9a Rita Brochado	06Ben-B	CAOV	3	30,0	9o Pedro Sousa	05Inf	GJVP	7	09,0
10a Cloe Ferreira	07Ben-B	IND-VC	3	37,0	10o Alexandre Malheiro	05Inf	OGADD	7	23,0
11a Mara Mesquita	07Ben-B	IND-VC	3	39,0	11o Edgar Vital	05Inf	IND-VC	7	48,0
12a Rita Traila	06Ben-B	OGADD	4	04,0					
13a Sara Simoes	06Ben-B	CYCL	4	08,0					
14a Anita Lopes	07Ben-B	CYCL	4	16,0	Milha Urbana		(Fem)		

Figura 5 – Dados Mal Analisados por PDFTables

No que diz respeito ao preço, ao contrário de “Tabula” que é gratuita, existe um custo de 2 a 3 cêntimos de dólar por cada página analisada após uma oferta de 25 páginas gratuitas.

“PDFix” é uma ferramenta que permite extrair dados de ficheiros PDF. Esta ferramenta é capaz de reconhecer a estrutura de um documento, detetar a ordem de leitura, detetar tabelas, linhas colunas e elementos duma hierarquia (http://pdfix.net. Acedido em 13 de Julho de 2017). No entanto, testando o seu conversor PDF para HTML é possível observar alguns erros de análise, como duas colunas no mesmo campo (ver figura 6) e duas tabelas assumidas como uma única (ver figura 7).

Figura 6 – Má Deteção de Coluna Com o PDFix

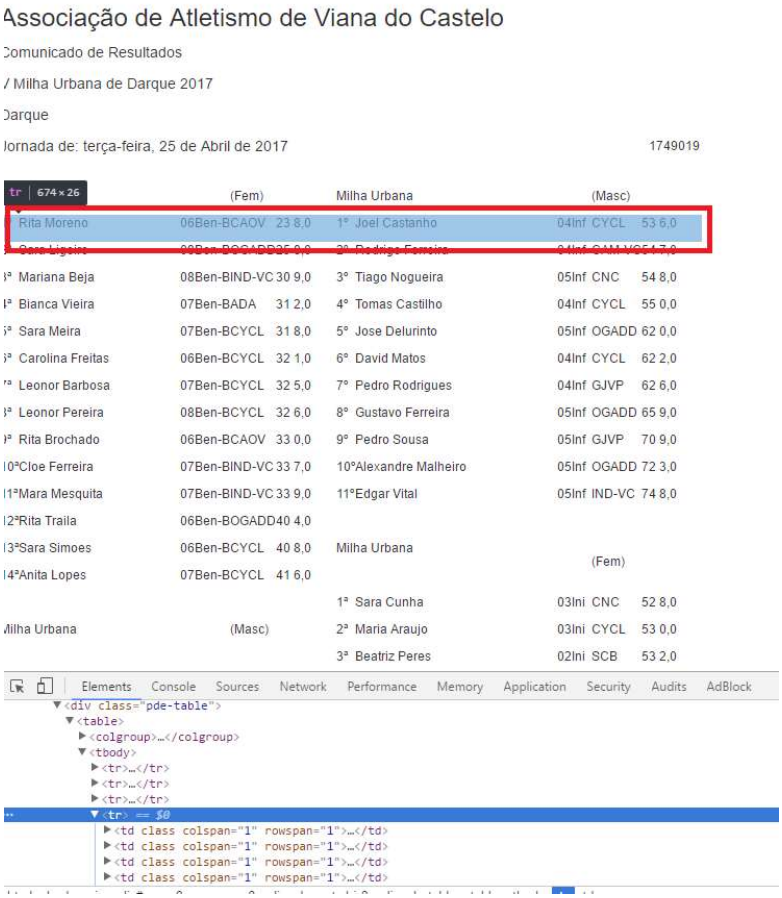


Figura 7 – Má Deteção de tabela e Coluna Com PDFix

A ferramenta "PDFix" tem uma API que pode ser acedida programaticamente, tornando possível escalar o processo. No entanto, o preço de uma licença varia entre os 490 a 2270 dólares, dependendo dos extras.
Sendo assim, e uma vez que não foi encontrada nenhuma das ferramentas que satisfaça todas as necessidades encontradas, resolvemos criar uma ferramenta. A ferramenta criada é apresentada no subcapítulo 4.3.

3.4 Uniformização e Limpeza dos Dados

A maior parte dos dados, mesmo estando limpos, possuem problemas de leitura e de consistência, especialmente se provierem de fontes diferentes. Sendo assim, é pouco provável a sua junção sem antes os formatar e uniformizar (Kazil, J. & Jarmul, K., 2016). Para esse efeito, foram consideradas para análise as três ferramentas comparadas na tabela 4. As ferramentas foram comparadas nos seguintes aspetos: capacidade de prever próximas transformações com base nas existentes, existência de algoritmos de normalização de dados (*clustering*), o limite máximo de dados analisados ao mesmo tempo, se possui ou não uma API programável, a licença, interface gráfica e preço.

	FuzzyWuzzy	**OpenRefine**	**TrifactaWrangler**
Previsão de Transformações	Não	Não	Sim
Clustering	Não	Sim	Não
Limite de Dados	Limitado apenas à RAM existente	Cerca de 800 000 entradas	100 MB
Escalabilidade (API)	Sim	Não	Não
Licença	GPL-2	BSD	Proprietária
Interface Gráfica	Não	Sim	Sim
Preço	Gratuito	Gratuito	Gratuito

Tabela 4 – Comparação de Ferramentas de Limpeza de Dados

"Fuzzywuzzy" é um módulo desenvolvido na linguagem *Python* capaz de comparar *strings* ente si pela sua semelhança ou diferença (Kazil, J. & Jarmul, K., 2016). Apesar de não ter um limite de dados, de permitir um processo automático e de poder ser usada em conjunto com as ferramentas "Pandas" ou "Regex", pode muitas vezes falhar

na sua análise, pois existem caso onde a escrita é semelhante e o significado é diferente e vice-versa.

As ferramentas "OpenRefine" e "TrifactaWrangler" seguem uma abordagem diferente. Ambas possuem uma interface gráfica que permite tratar dos dados de uma só vez, mas o processo terá de ser repetido sempre que novos dados necessitem de tratamento, este processo tem o nome de "*wrangling*". À medida que a análise de dados se tornou mais sofisticada ao longo do tempo, tem havido poucas melhorias na parte mais monótona do fluxo da sua obtenção: entrada, formatação e limpeza de dados… Isto significa que pessoas especializadas ficam mais "presas" no trabalho banal de limpeza de dados do que em exercer a sua especialidade, ao passo que utilizadores menos técnicos ficam barrados desnecessariamente (Kandel et al., 2011).

O "TrifactaWrangler" é capaz de efetuar previsões com base nas transformações efetuadas pelo utilizador. No entanto, a versão gratuita apenas permite trabalhar com um máximo de 100 MB (https://www.trifacta.com/products/editions. Acedido em 16 de Janeiro de 2018).

O "OpenRefine" é um projeto *open source* (código aberto) iniciado em 2009 pela empresa Metaweb com o nome de "FreebaseGridworks". Em 2010 a empresa foi adquirida pela Google, que mudou o nome do projeto para "Google Refine". Em 2012 a Google abandonou o apoio do projeto dando-lhe o nome de "OpenRefine" onde qualquer pessoa é livre de contribuir para o seu desenvolvimento (Mitchell, R., 2015).

Conclui-se que "OpenRefine" é a melhor solução porque é capaz de limpar os dados de forma rápida e fácil até mesmo por não programadores (Mitchell, R., 2015). Apesar de não efetuar a previsão de transformações permite trabalhar com uma maior quantidade de dados do que o "Trifacta wrangler" (cerca de 800 mil entradas) e possui bons algoritmos capazes de realizar o *clustering* de *strings* com o mesmo significado de forma semiautomática.

3.5 Ferramentas de Análise de Dados

Ferramentas de análises de dados são necessárias para agregar campos das APIs com dados geográficos e meteorológicos aos já existentes e introduzi-los num DW. Na tabela 5 existe uma comparação das ferramentas consideradas de maior interesse.

	Jupyter	Pandas	Qlik View Personal Edition	Qlik Sense Cloud	Dataiku
ETL	Não	Programável	Sim	Sim	Sim
Escalabilidade (API)	Não	Sim	Não	Não	Não
Licença	BSD-3	BSD-3	Proprietária	Proprietária	Proprietária
Interface Gráfica	Sim	Não	Sim	Sim	Sim
Preço	Gratuito	Gratuito	Versão Limitada Gratuita	Versão Limitada Gratuita (Alojamento Incluído)	Gratuito (Versão Sem Alojamento)

Tabela 5 – Comparação de Ferramentas de Análise de Dados

"QlikViewPersonalEdition", "QlikSenseCloud" e Dataiku são ferramentas de análise de dados que para além de permitirem operações de DM e BI, também possuem ETL integrado, no entanto, é possível obter um maior controlo sobre as operações de ETL recorrendo ao módulo *Python "*Pandas" e facilitar a sua utilização através da aplicação "Jupyter".

"Jupyter" (anteriormente conhecido como IPython Notebook) é uma aplicação Web que permite "apresentar e executar *Python* interactivamente", ajudando os programadores a "escrever código capaz de interagir com o utilizador final"(Reitz, K. & Schlusser, T., 2016). Além disso, é também uma boa ferramenta que permite partilhar código e gráficos entre vários utilizadores, através da *internet,* para exploração de dados (Kazil, J. & Jarmul, K., 2016). Cada trabalho é realizado num chamado "bloco de notas". Estes "blocos de notas" combinam a facilidade de utilização de um *browser* com as caraterísticas interativas do IPython. Além disso, são uma excelente forma de apresentar o fluxo de trabalho localmente ou partilhado e de grande utilidade para executar localmente à medida que os dados de exploração e análise são iterados (Kazil, J. & Jarmul, K., 2016).

"Pandas", derivado do nome "*Panel Data*", é um módulo de manipulação de dados que possui funções de grande utilidade para indexar, aceder, agrupar e juntar dados com grande facilidade (Reitz, K. & Schlusser, T., 2016), permitindo não só efetuar operações de ETL como também aumentar os dados já existentes numa *Data frame* com outros provenientes de APIs externas.

3.6 APIs Externas

Recorreu-se a 2 APIs para obter novos dados com base nos já existentes:

- A Google Maps API, para obter dados geográficos (coordenadas GPS e altitude) das provas. A maior parte das APIs da Google são gratuitas e bastante liberais quanto à sua utilização (Mitchell, R., 2015);
- A API da *Weather Underground*, para obter dados do estado do tempo a partir das coordenadas GPS por sua vez obtidas com a Google Maps API. A *Weather Underground* é uma empresa que tem partilhado os seus dados meteorológicos de todo o mundo ao público desde 1993. O seu serviço gratuito permite 500 pedidos por dia e 10 por minuto (https://www.wunderground.com. Acedido em 30 de Janeiro de 2018).

A API do Facebook (https://www.facebook.com) foi brevemente considerada, mas não testada pela potencial complexidade na obtenção dos dados. A sua utilização seria para obter informação pessoal do atleta (caso esteja disponível publicamente) através da API do Facebook, recorrendo à que já está na nossa posse: nome, data de nascimento, clube de atletismo e distrito onde possivelmente vive.

Outras fontes de potencial utilidade seriam sites de saúde e fitness como o "SportsTracker" (http://www.sports-tracker.com), "Endomondo" (https://www.endomondo.com) e "RunKeeper" (https://runkeeper.com), repositórios de rotas desportivas como "GPSies" (https://www.gpsies.com) e "Komoot" (https://www.komoot.com) e serviços de cronometragem de eventos desportivos como a "Chrono" (http://chrono.pt).

3.7 Ferramentas de BI e DM

Para realizar BI e DM foram comparadas as ferramentas da tabela 6.

	Orange Canvas	Metabase	QlikSenseCloud	Apache Superset	Tableau Public	Redash
ETL	Não	Não	Sim	Não	Operações Simples	Não
DM	Sim	Não	Não	Não	Sim	Não
BI	Não	Sim	Sim	Sim	Sim	Sim
Interface Gráfica	Sim	Sim	Sim	Sim	Sim	Sim
Interface Web	Não	Sim	Sim	Sim	Sim	Sim
Licença	GPL-3	AGPL-3	Proprietária	APL-2	Proprietária	BSD-2
Preço	Gratuito	Gratuito	Gratuito	Gratuito	Gratuito	Gratuito

Tabela 6 – Comparação de Ferramentas de Análises de Dados (BI e DM)

A ferramenta "Orange Canvas" foi a ferramenta selecionada para DM porque é um software gratuito, de código aberto e possui bastantes métodos de visualização (Demšar, J., 2010). A sua interface gráfica possui *widgets* que podem ser adicionados e ligados entre si para criar um fluxo de dados (Demšar, J., 2010) facilitando o processo de DM. É possível alterar os parâmetros de visualização diretamente dos gráficos e tabelas tornando o fluxo de dados interativo.

Para o BI optou-se pela ferramenta "Metabase" porque é uma ferramenta de código aberto e de fácil utilização. Funciona com os 3 principais sistemas operativos (Linux, Windows e Mac) e pode ser utilizado num servidor ou em formato *standalone* num portátil. A ferramenta é extremamente prática, fácil de usar e a sua instalação é extremamente simples, algo pouco comum para soluções de BI (https://www.metabase.com. Acedido em 21 de Janeiro de 2018).

3.8 Bases de Dados Desportivas Centralizadas

Não existem muitas bases de dados disponíveis relacionadas com atletismo. As principais observadas e comparadas com o DW proposto nesta investigação (tabela 7), são as seguintes:

- all-athletics.com que cobre os rankings da maior parte dos países, mas apenas a nível nacional. O custo de uma subscrição de 6 meses é de 43 euros (http://www.all-athletics.com/pt. Acedido em 13 de Julho de 2017);

- ope.ed.gov, um repositório dos Estados Unidos com uma vasta quantidade de dados sobre resultados desportivos em escolas (https://ope.ed.gov/athletics. Acedido em 13 de Julho de 2017);
- thepowerof10.info, um website que reúne informação sobre atletas Britânicos (http://thepowerof10.info. Acedido em 13 de Julho de 2017).

A base de dados idealizada e construída no projeto apresentado neste documento, é composta por dados de provas de atletismo realizadas em Portugal tanto ao nível nacional como ao nível distrital.

	all-athletics.com	ope.ed.gov	thepowerof10.info	DW do Projeto
Outros Desportos	Não	Sim	Não	Não
Países Abrangidos	A Maior Parte dos Países	Estados Unidos	Reino Unido	Portugal
Nível Geográfico	Nacional	Nacional	Regional	Distrital
Disponibilidade de Ficheiro	Não	Sim	Não	Sim
Número de Campos	10 - 15	150 - 160	10 - 15	40
Número de Entradas (Aproximação)	5 Milhões	20000	Indeterminado	1 Milhão
Performance Individual de Cada Atleta	Sim	Não	Sim	Sim
Preço	43 Euros (6 Meses)	Grátis	Grátis	Grátis

Tabela 7 Comparação de Bases de Dados Centralizadas de Atletismo

Que seja do nosso conhecimento, não existe ainda um DW centralizado de resultados de prova de atletismo de atletas portugueses. No entanto, seria interessante reunir essa vasta quantidade de informação num DW para uma posterior análise. Como "prova" de que um DW semelhante ainda não existe, e de que há uma procura, temos um comentário, encontrado nas redes sociais, feito pela associação nacional de atletismo de veteranos que podemos ver na figura 8.

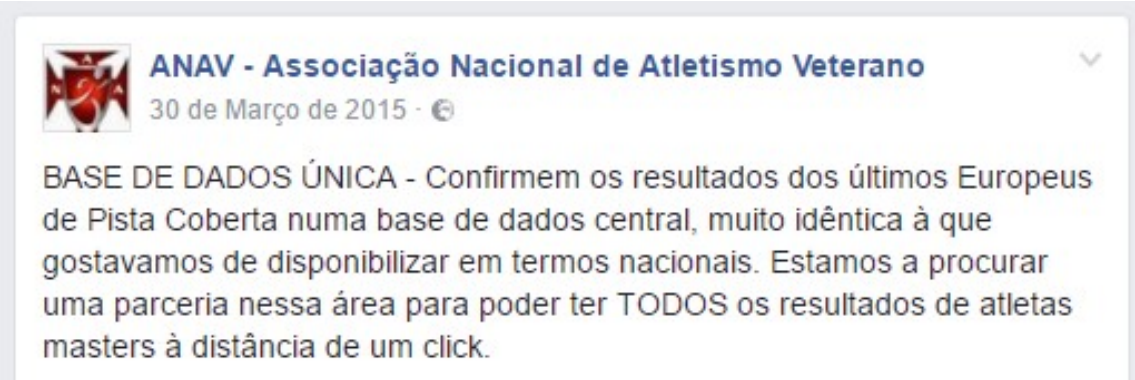


Figura 8 – Procura de um DW com Resultados de Atletismo de Atletas Veteranos

4. Design e Desenvolvimento

Neste capítulo apresentam-se e explicam-se as diversas fases do trabalho criado durante esta investigação.

4.1 Primeira Fase – Web Scraping

A primeira fase envolve construir um *Web scraper* para cada associação distrital de atletismo, encontrar os *links* de todos os ficheiros com os resultados e carregá-los.
A ferramenta utilizada depende do tipo de *Website*, se não utilizar JavaScript para preencher o conteúdo da página é utilizado o "Scrapy 1.5", caso contrário recorre-se ao "Selenium Webdriver" para simular um *Web browser* real. O fluxo desta seleção pode ser visto na figura 9.

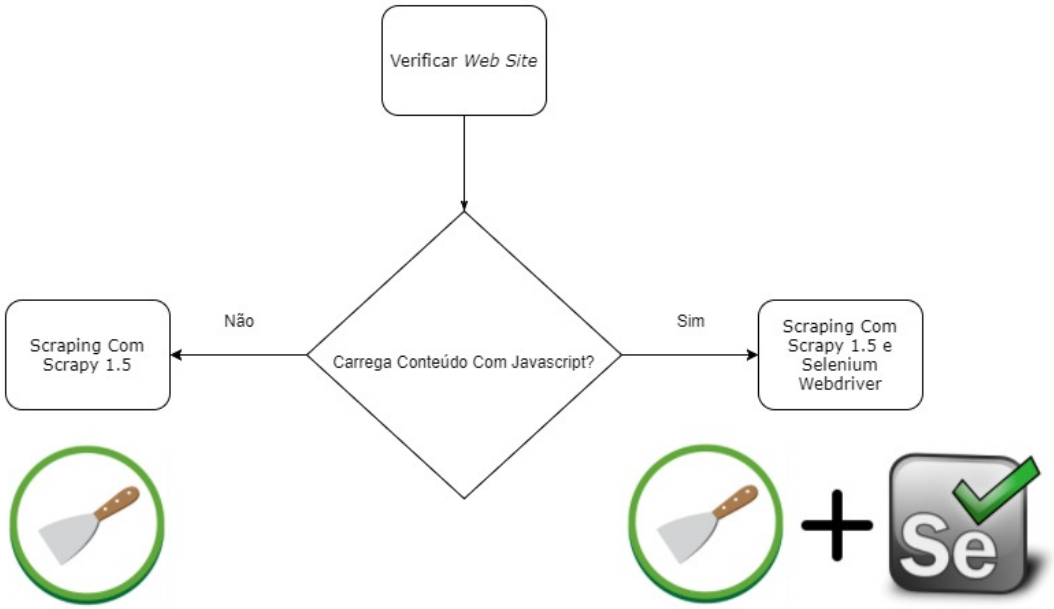


Figura 9 – Procedimento para Obter os Ficheiros de Resultados

Para a instalação do "Scrapy 1.5" e "Selenium Webdriver" no Windows basta ter a distribuição "Python" "Anaconda" instalada e, na linha de comandos escrever:

```
conda install -c conda-forge scrapy
conda install -c conda-forge selenium
```

Para criar a estrutura inicial do projeto com o "Scrapy"na linha de comandos introduz-se:

```
scrapy startproject AtletismoSpider
```

Isto cria a diretoria "AtletismoSpider" e o seguinte conteúdo:

```
AtletismoSpider/
    scrapy.cfg
    AtletismoSpider/
        __init__.py
        items.py
        middlewares.py
        pipelines.py      # project pipelines file
        settings.py       # project settings file
        spiders/  # a directory where you'll later put your spiders
            __init__.py
```

Em “items.py” é o ficheiro onde são definidos os campos a extrair dos websites. Neste caso apenas existe 1 campo que é o *link* dos ficheiros de resultados.

```
from scrapy import Item, Field

# definir uma classes para os itens de ficheiro
class Ficheiro(Item):
    # definir os campos dos itens
    link = Field()
```

Na pasta “spiders” são programados os *Web crawlers*. Tomando o distrito de Aveiro como exemplo temos o seguinte código:

```
from scrapy.spiders import CrawlSpider, Rule
from scrapy.linkextractors import LinkExtractor
from scrapy.selector import Selector
from AtletismoSpider.items import Ficheiro

from urllib.parse import urljoin

class MySpider(CrawlSpider):
    download_delay = 0.25

    name = 'aveiro'
    allowed_domains = ['aaaveiro.pt']
    start_urls = ['http://www.aaaveiro.pt/resultados.aspx',
                  'http://www.aaaveiro.pt/ranking.aspx',
                  'http://www.aaaveiro.pt/recordes.aspx']

    rules = (Rule(LinkExtractor(
                  restrict_xpaths='//td[@class="textodir"]/a'),
                  follow=True,
                  callback='parse_item',),)

    def parse_item(self, response):
        hxs = Selector(response)
        links = hxs.xpath('//h3[@class="sub3"]/a/@href')

        items = []
        for link in links:
            item = Ficheiro()
            item['link'] = urljoin(response.url,
                                   link.extract().strip())
            items.append(item)
```

```
        return items
```

De destacar algumas variáveis no código:

- download_delay – tempo de intervalo entre cada página carregada;
- name – nome dado ao *Web scraper* para futura evocação;
- allowed_domais – a análise apenas será feita dentro destes domínios da *internet*;
- start_urls – lista de uniform resource locators (URLs) onde o *crawling* terá início;
- rules – definições das regras de análise, onde são definidas as *extensible markup languages* (XML) *Paths* (XPaths), dos *links* onde a continuação da análise é permitida;
- item['link'] – o campo "link" da classe "Ficheiro" definido em "items.py".

Quando um distrito apresenta o seu conteúdo após a página já ter sido carregada o "Selenium Webdriver" é utilizado, como é o caso do distrito de Viseu, apresentado no exemplo seguinte:

```
from scrapy.spiders import Spider
from AtletismoSpider.items import Ficheiro

from urllib.parse import urljoin
from selenium import webdriver
from time import sleep


class MySpider(Spider):
    download_delay = 0.25

    name = 'viseu'
    allowed_domains = ['aaviseu.wixsite.com']
    start_urls = [
                  'http://aaviseu.wixsite.com/aaviseu/20142015',
                  'http://aaviseu.wixsite.com/aaviseu/20132014'
                  ]

    def __init__(self):
        # inicia a sessão do webdriver como o browser chrome
        self.browser = webdriver.Chrome('./chromedriver.exe')

    def __del__(self):
        # fecha as janelas dos browsers
        # finaliza a sessão do webdriver de forma graciosa
        self.browser.quit()

    def parse(self, response):
        item = Ficheiro()

        browser = self.browser
        browser.get(response.url)
        sleep(2.5)
        links = browser.find_elements_by_xpath(
            '//div[@id="PAGES_CONTAINER"]//a');

        for link in links:
            link = link.get_attribute('href')
```

```
            item['link'] = urljoin(response.url, link.strip())
            yield item
```

Para que seja possível o "Selenium" comandar um *Web browser* é necessário indicar o driver do *browser* escolhido. No caso do projeto desenvolvido na investigação aqui apresentada, foi escolhido o do Google Chrome.

Sempre que um URL de um ficheiro com resultados de atletismo é encontrado, é necessário efetuar o seu download. O código que deve ser escrito dentro do ficheiro "pipelines.py" onde ficam as operações necessárias após a obtenção dos dados é o seguinte:

```
# -*- coding: utf-8 -*-

# Define os pipelines dos items
# Adicionar os pipelines ao ITEM_PIPELINES em definições (settings)
# Ver: http://doc.scrapy.org/en/latest/topics/item-pipeline.html

from pathlib import Path                # criar diretorias
from urllib.request import urlopen      # abrir ficheiros da net
from hashlib import md5                 # checksum do ficheiro
from urllib.request import urlretrieve  # abrir ficheiros da net
from os.path import isfile              # verificar se ficheiro existe
from urllib import parse                # para criar URI válido
from mimetypes import guess_extension   # obter extensão do ficheiro


class DuplicadosPipeline(object):
    def process_item(self, item, spider):
        # criar diretoria se ainda não existir
        Path(spider.name).mkdir(parents=True, exist_ok=True)

        # criar ficheiro se não existir
        try:
            open(spider.name + '/carregados.txt', 'x')
        except FileExistsError:
            pass

        # obter links dos ficheiros já carregados
        links_guardados = []
        with open(spider.name + '/carregados.txt', 'r') as f:
            dados = f.read()
            links_guardados = dados.split('\n')
            f.flush()

        # converter espaços em branco etc do URI
        # separar nome de ficheiro do link
        url_dividido = list(parse.urlsplit(item['link']))
        # descodificar link caso não esteja
        url_dividido[2] = parse.unquote(url_dividido[2])
        # codificar link caso não esteja
        url_dividido[2] = parse.quote(url_dividido[2])
        # voltar a juntar nome de ficheiro e link
        link_limpo = parse.urlunsplit(url_dividido)

        # verificar se o link obtido pelo web scraper é duplicado
        # se não for, guardar o respetivo ficheiro do link
        # e acrescentar o link aos carregados se gravado corretamente
```

```
        if link_limpo in links_guardados:
            return 'ficheiro ' + link_limpo + ' é duplicado.'
        else:
            # obter conteúdo do ficheiro de resultados
            ficheiro = urlopen(link_limpo)
            conteudo = ficheiro.read()
            # resumir o conteúdo do ficheiro (checksum)
            conteudo_md5 = md5(conteudo).hexdigest()

            # obter o tipo de conteúdo
            tipo_ficheiro = ficheiro.headers['content-type']
            # obter extensão do ficheiro
            extensao = guess_extension(tipo_ficheiro)
            # carregar ficheiro pelo seu link e guardá-lo
            nome_ficheiro = spider.name + '/' + conteudo_md5 +
extensao
            urlretrieve(link_limpo, nome_ficheiro)

            # se ficheiro foi carregado com sucesso
            ficheiro_carregado = Path(nome_ficheiro)
            if ficheiro_carregado.is_file():
                # guardar link de ficheiro guardado
                with open(spider.name + '/carregados.txt', 'a') as f:
                    f.write(link_limpo + '\n')
                    f.flush()
                    # ficheiro é guardado com sucesso
                    return 'ficheiro ' + link_limpo + ' foi guardado.'
            # senão dizer que ficheiro não foi guardado
            else:
                return 'ficheiro ' + link_limpo + ' não foi guardado.'
```

Para que a classe tenha efeito é necessário acrescentar ao ficheiro “settings.py” o código:

```
ITEM_PIPELINES = {
    'AtletismoSpider.pipelines.DuplicadosPipeline': 300,
}
```

Onde “300” é um número de 0 a 1000 que define a ordem pela qual as classes são evocadas. Neste caso como apenas existe uma classe foi introduzido um número aleatório de “300”.

O código da classe “DuplicadosPipeline” do ficheiro “pipelines.py” verifica se os ficheiros extraídos são iguais a outros ficheiros extraídos anteriormente. Uma vez que comparar todo o conteúdo de cada ficheiro seria uma tarefa muito pesada, aplica-se o algoritmo de *hashing message digest* 5 (MD5) ao conteúdo de um ficheiro, antes de ser carregado e compara-se o resultado à lista de *hashes* de ficheiros já carregado. Caso o valor já exista, o documento é descartado, caso contrário, o documento é armazenado e o novo valor *hash* é guardado na lista que armazena o *hash* dos ficheiros já carregado. Um exemplo simplificado do fluxo do código pode ser observado na figura 10.

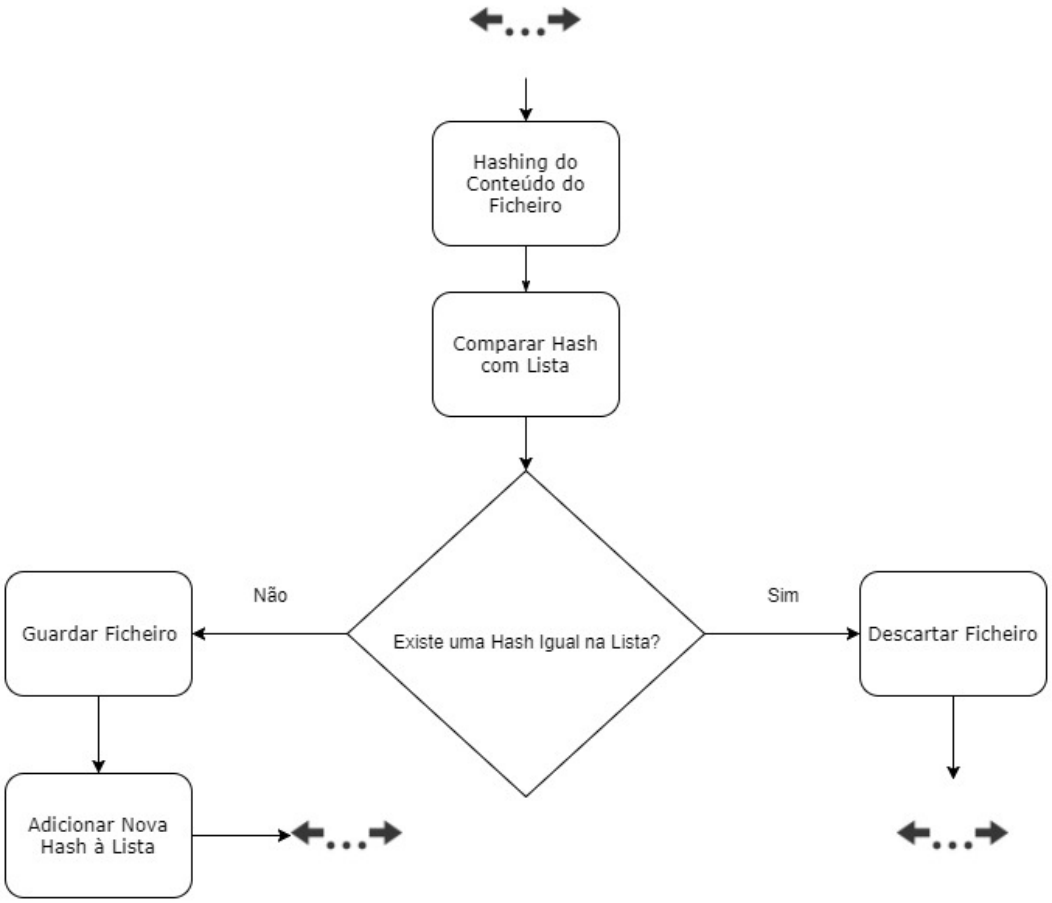


Figura 10 – Verificação de Ficheiros Duplicados

Comparar os ficheiros pelas suas datas pode ser enganador porque o seu autor pode publicar múltiplos ficheiros no mesmo dia ou introduzir uma data anterior à última análise.

Para obter os novos ficheiros de um dos distritos basta evocar o seguinte comando, na diretoria principal do projeto criado:

```
scrapy crawl aveiro
```

Sendo "aveiro" o nome do *Scraper* a ser evocado.

4.2 Segunda Fase – Verificação do Conteúdo

Mesmo comparando os *links* originais de cada ficheiro não impede totalmente que existam duplicados, uma vez que, o administrador do *Web site* pode resolver substituir um deles mantendo o seu URL original, a lista de *links* já carregados pode ser apagada acidentalmente ou outros ficheiros podem provir de fontes diferentes. Uma vez que, comparar todo o conteúdo de cada ficheiro seria uma tarefa muito pesada, aplica-se o

algoritmo *hashing message digest* 5 (MD5) ao conteúdo de cada ficheiro e define-se o resultado como o seu novo nome, sendo praticamente impossível ter a mesma chave para ficheiros de conteúdo diferente. Desta forma é virtualmente impossível repetir ficheiros, pois o próprio sistema operativo encarrega-se de impedir a existência de nomes iguais na mesma diretoria.

Na primeira fase do trabalho apenas são verificados os *links* repetido e apenas nesta fase é verificado se existem ficheiros de conteúdo igual. Desta forma não há necessidade de ter todos os ficheiros juntos durante o processo de *Web scraping* para verificar a maior parte das repetições.

Para todo este processo foi criado um script onde a seguinte função, desenvolvida em *Python*, seria aplicada a cada ficheiro:

```
# obter tipo de ficheiro
def hash_ficheiro(caminho_ficheiro):
    # obter conteúdo do ficheiro de resultados
    with open(caminho_ficheiro, 'rb') as f:
        conteudo = f.read()
        # resumir o conteúdo do ficheiro (checksum)
        conteudo_md5 = md5(conteudo).hexdigest()

    # obter extensão do tipo de ficheiro
    tipo_ficheiro = guess_type(caminho_ficheiro)[0]
    try:
        extensao = guess_extension(tipo_ficheiro)
    except AttributeError:
        return ('Nome não alterado, tipo não encontrado: '
                + caminho_ficheiro)

    # renomear ficheiro com o md5 do seu conteúdo
    # ou remover ficheiro caso já exista um com o mesmo nome
    diretoria = dirname(caminho_ficheiro)
    destino = join(diretoria, conteudo_md5 + extensao)
    try:
        if caminho_ficheiro == destino:
            pass
        else:
            rename(caminho_ficheiro, destino)
    except FileExistsError:
        remove(caminho_ficheiro)
        return 'Ficheiro apagado, nome repetido: ' + caminho_ficheiro
```

Outra tarefa importante na verificação do conteúdo é a uniformização do tipo de ficheiro para um formato .txt. Primeiro todos os formatos (excepto .txt e .pdf) são programaticamente convertidos em .pdf utilizando o "LibreOffice" pela linha de comandos. Em seguida todos os PDFs são convertidos em texto, mantendo a estrutura do conteúdo o melhor que for possível recorrendo à ferramenta "pdftotext". Um resumo do fluxo do trabalho pode ser observado na figura 11.

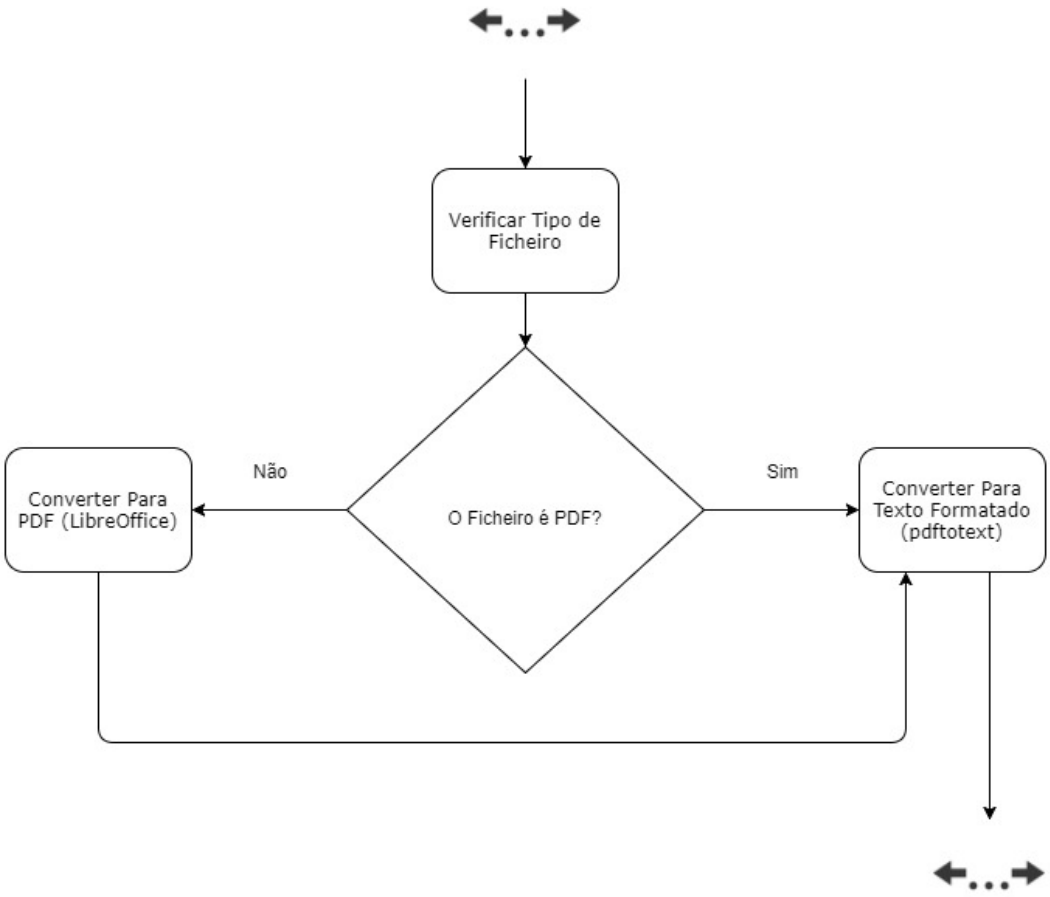


Figura 11 – Fluxo de Trabalho Para a Normalização do Tipo de Ficheiro

Do código criado, as funções mais importantes são “converter_pdf” e “converter_txt” (apresentadas em seguida) que são responsáveis por converter ficheiros para PDF e texto plano, respetivamente.

```
# converter ficheiros para pdf
# dependências:
# - Libre Office
# - definir variável global
# - OpenJDK 8 ou maior
def converter_pdf(diretoria, tipo_ficheiro):
    for caminho, sub_diretorias, ficheiros in walk(diretoria):
        caminho_ficheiros = [join(caminho, f) for f in ficheiros
                             if f.endswith(tipo_ficheiro)]
        for caminho_ficheiro in caminho_ficheiros:
            call(['soffice', '--convert-to', 'pdf', caminho_ficheiro,
                  '--outdir', diretoria])
            print(caminho_ficheiro + ' convertido para pdf')
    return True

# converte ficheiros de pdf para txt formatado
# dependências:
# - pdftotext
```

```
def converter_txt(diretoria):
    for caminho, sub_diretorias, ficheiros in walk(diretoria):
        caminho_ficheiros = [join(caminho, f) for f in ficheiros]
        for caminho_ficheiro in caminho_ficheiros:
            call(['pdftotext', '-layout', caminho_ficheiro])
            print(caminho_ficheiro + ' convertido para txt')
    return True
```

4.3 Terceira Fase – Desenvolvimento do *PositionParser*

A terceira fase envolve a extração dos dados dos ficheiros de texto para CSV ao mesmo tempo que é descoberta a relação entre os valores e os campos. Como não foi encontrada uma ferramenta suficientemente adequada para este trabalho, foi construído um módulo para esse efeito. O módulo *PositionParser* criado no âmbito desta investigação é um protótipo que dá suporte às seguintes atividades:

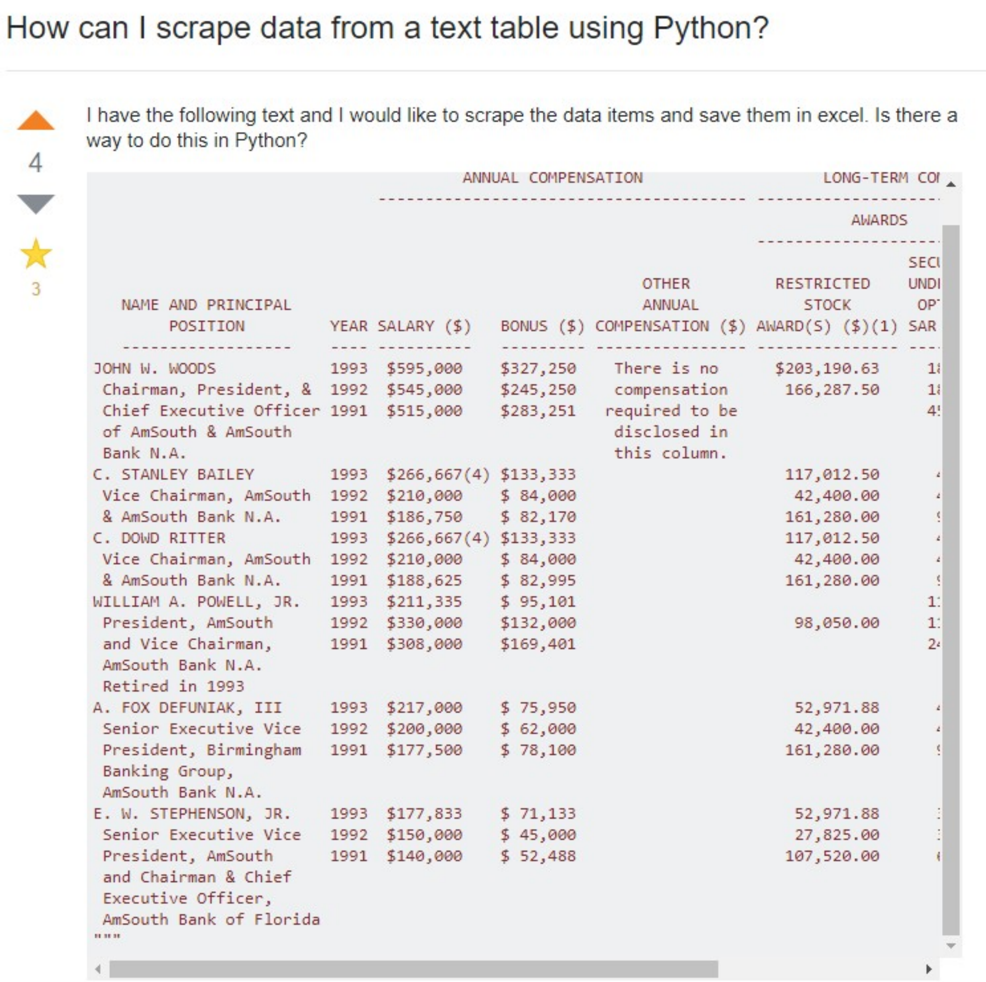
1. "Tokenizar" os dados, segmentando o conteúdo original em pequenas partes representando uma palavra com o valor da linha e coluna onde esta se inicia no texto e um rótulo de identificação do valor da palavra;
2. Definir os nomes dos campos, marcando ou alterando os valores dos rótulos dos *tokens*;
3. Visualizar os dados "tokenizados", reconstruindo os tokens novamente em texto para ter uma ideia sobre o que está rotulado e segmentado;
4. Definir tabelas e ordem de leitura, decompondo o grupo de tokens em segmentos mais pequenos;
5. Extrair os valores e definir a hierarquia dos campos, tendo em conta os diferentes graus de abstração que estes podem ter entre si e guardar os dados num ficheiro *comma separated value* (CSV) ou *JavaScript Object Notation* (JSON).

Apesar de não ter uma interface gráfica, pode ser acedido programaticamente permitindo automatizar, de certa forma o processo de análise. Através de testes efetuados, no âmbito desta investigação, a outras ferramentas de extração de conteúdo, pode-se concluir que quando a identificação de tabelas, campos, ordem de leitura e hierarquia de campos é feita automaticamente a propensão para erros é tão grande que deveria ser considerada uma abordagem manual, permitindo que um algoritmo mais estável possa ser desenvolvido para cada caso (onde as tabelas estão estruturadas de igual modo). A abstração do trabalho viria antes do uso de atributos amplos (comuns para qualquer palavra do texto) como expressões regulares (Regex), posição absoluta, posição relativa e semelhança entre palavras.

4.4 Terceira Fase – Testes do *PositionParser*

Como forma de testar o protótipo, foi aplicado na resolução de um problema colocado no "*Stackexchange*" (https://stackoverflow.com), um Web site de perguntas e respostas, onde um utilizador pede uma solução para extrair os dados de um texto simples, demonstrado na figura 12 (https://stackoverflow.com/questions/5873969/

how-can-i-scrape-data-from-a-text-table-using-python. Acedido em 8 de Fevereiro de 2018).




```
                                       ANNUAL COMPENSATION                    LONG-TERM CO
                              -------------------------------------- -------------------
                                                                              AWARDS
                                                                     -------------------
                                                                                    SEC
                                                           OTHER       RESTRICTED   UND
  NAME AND PRINCIPAL                                       ANNUAL        STOCK       OP
       POSITION             YEAR SALARY ($)    BONUS ($) COMPENSATION ($) AWARD(S) ($)(1) SAR
  -----------------         ---- ----------    --------- --------------- -------------- ---
JOHN W. WOODS               1993  $595,000     $327,250    There is no     $203,190.63    1
 Chairman, President, &     1992  $545,000     $245,250    compensation     166,287.50    1
 Chief Executive Officer    1991  $515,000     $283,251   required to be                  4
 of AmSouth & AmSouth                                      disclosed in
 Bank N.A.                                                 this column.
C. STANLEY BAILEY           1993  $266,667(4)  $133,333                     117,012.50
 Vice Chairman, AmSouth     1992  $210,000     $ 84,000                      42,400.00
 & AmSouth Bank N.A.        1991  $186,750     $ 82,170                     161,280.00
C. DOWD RITTER              1993  $266,667(4)  $133,333                     117,012.50
 Vice Chairman, AmSouth     1992  $210,000     $ 84,000                      42,400.00
 & AmSouth Bank N.A.        1991  $188,625     $ 82,995                     161,280.00
WILLIAM A. POWELL, JR.      1993  $211,335     $ 95,101                                   1
 President, AmSouth         1992  $330,000     $132,000                      98,050.00    1
 and Vice Chairman,         1991  $308,000     $169,401                                   2
 AmSouth Bank N.A.
 Retired in 1993
A. FOX DEFUNIAK, III        1993  $217,000     $ 75,950                      52,971.88
 Senior Executive Vice      1992  $200,000     $ 62,000                      42,400.00
 President, Birmingham      1991  $177,500     $ 78,100                     161,280.00
 Banking Group,
 AmSouth Bank N.A.
E. W. STEPHENSON, JR.       1993  $177,833     $ 71,133                      52,971.88
 Senior Executive Vice      1992  $150,000     $ 45,000                      27,825.00
 President, AmSouth         1991  $140,000     $ 52,488                     107,520.00
 and Chairman & Chief
 Executive Officer,
 AmSouth Bank of Florida
"""
```

Figura 12 – Questão do Stackexchange

Este tipo de análise não é uma tarefa simples, como é possível ver na resposta dada por um utilizador (ver figura 13). Mas uma solução para este padrão em particular foi apresentada por outro utilizador da comunidade para extrair apenas 1 campo, tal como demonstrado na figura 14.

2

You need to write a series of generators to make successive passes at the data to reduce noise and complexity.

This is **not** an easy problem to solve in any programming language.

```
def strip_top( source_text ):
    src= iter( source_text )
    for line in src:
        if line.rstrip().startswith("AWARDS"):
            next( src )
            break
    for line in src:
        yield line

def columnize( source_text ):
    """Assumes strip_top or similar to remove confusing extra headers"""
    for line in src:
        yield line[0:24], line[25:30], ... for each coumn

def collapse_headers( source_text ):
    """Assumes columnize( strip_top())."""
    src= iter( source_text )
    headings= [ [] for i in range(9) ]
    for line in src:
        if line[0] == "------------------":
            break
        for col in range(9):
            headings[col].append(line[col].strip())
    yield [ " ".join(h) for h in headings ]
    for line in src:
        yield line

etc.
```

Then, your "main" program assembles these transformations into a pipeline.

```
with open("some file","r") as text:
    for line in collapse_headers( columnize( strip_top( text ) ) ):
        # further cleanup?
        # actual processing
```

This allows you to "tweak" each piece of the transformation sequence separately.

share edit flag

answered May 3 '11 at 19:10

S.Lott

281k • 56 • 396 • 678

Figura 13 – Resposta Afirmando que uma Análise de Tabelas Sem Estrutura não é Simples

Here is some code to get you started:

```
text = """JOHN ..."""  # text without the header

# These can be inferred if necessary
cols = [0, 24, 29, 39, 43, 52, 71, 84, 95, 109, 117]

db = []
row = []
for line in text.strip().split("\n"):
    data = [line[cols[i]:cols[i+1]] for i in xrange((len(cols)-1))]
    if data[0][0] != " ":
        if row:
            db.append(row)
        row = map(lambda x: [x], data)
    else:
        for i, c in enumerate(data):
            row[i].append(c)
print db
```

This will produce an array with an element per person. Each element will be an array of all the columns, and that will hold an array of all the rows. This way you can easily access the different years, or do things like concatenate the person's title:

```
for person in db:
    print "Name:", person[0][0]
    print " ".join(s.strip() for s in person[0][1:])
    print
```

Will yield:

```
Name: JOHN W. WOODS
Chairman, President, & Chief Executive Officer of AmSouth & AmSouth Bank N.A.

Name: C. STANLEY ...
```

share edit flag

answered May 3 '11 at 19:11

Gustav Larsson

4,688 ● 1 ● 17 ● 44

Figura 14 – Resposta Apresentando uma Solução Para 1 dos Campos

Recorrendo ao *PositionParser*, a solução para extrair 1 campo, torna-se bastante mais simples sendo necessárias apenas 3 linhas de código, como se pode ver nas figuras 15 e 16. A continuidade do trabalho tornou também possível extrair todos os campos, definir a sua hierarquia e guardar os dados num ficheiro CSV. Especialmente se for utilizado em conjunto com o ambiente Jupyter, demonstrado nas figuras 17, 18, 19, 20 e 21.

Label Handling

```
tags = [('name', None, (0, 1), None)]
tokens = tag(tokens, tags)
tokens = spread(tokens, 'name', 'name', edge=0)
```

Figura 15 – Código para extrair os dados do Campo "Nome"

Module Import

In [1]:
```
from PositionParser import strip_blanks, tokenize, tag, tag_lines, reconstruct, spread, mirror, chop, retag, extract, save
from IPython.display import display
```

Tokenization

In [2]:
```
raw_file = open('stackexchange.txt', 'r')
corpus = raw_file.read()

print(corpus)

tokens = tokenize(corpus)
```

```
                                       ANNUAL COMPENSATION                  LONG-TERM COMPENSATION
                              --------------------------------------- -------------------------------------
                                                                         AWARDS               PAYOUTS
                                                                      -------------------------- ----------
                                                                                    SECURITIES
                                                           OTHER       RESTRICTED   UNDERLYING             ALL OTHER
   NAME AND PRINCIPAL                                      ANNUAL         STOCK      OPTIONS/      LTIP    COPMPENSA-
        POSITION          YEAR SALARY ($)    BONUS ($) COMPENSATION ($) AWARD(S) ($)(1) SAR'S (#) PAYOUTS($) TION($)(3)
   ------------------     ---- ----------    --------- ---------------- --------------- --------- ---------- ----------
JOHN W. WOODS             1993 $595,000     $327,250    There is no      $203,190.63      18,000               $ 29,295
 Chairman, President, &   1992 $545,000     $245,250    compensation      166,287.50      18,825    (2) Not    $ 29,123
 Chief Executive Officer  1991 $515,000     $283,251   required to be                     45,000   Applicable
 of AmSouth & AmSouth                                   disclosed in
 Bank N.A.                                              this column.
C. STANLEY BAILEY         1993 $266,667(4) $133,333                       117,012.50       4,500               $ 11,648
 Vice Chairman, AmSouth   1992 $210,000    $ 84,000                        42,400.00       4,800               $ 12,400
 & AmSouth Bank N.A.      1991 $186,750    $ 82,170                       161,280.00       9,750
C. DOWD RITTER            1993 $266,667(4) $133,333                       117,012.50       4,500               $ 13,566
 Vice Chairman, AmSouth   1992 $210,000    $ 84,000                        42,400.00       4,800               $ 12,920
 & AmSouth Bank N.A.      1991 $188,625    $ 82,995                       161,280.00       9,750
WILLIAM A. POWELL, JR.    1993 $211,335    $ 95,101                                       11,000               $124,548
 President, AmSouth       1992 $330,000    $132,000                        98,050.00      11,100               $ 22,225
 and Vice Chairman,       1991 $308,000    $169,401                                       24,000
 AmSouth Bank N.A.
 Retired in 1993
A. FOX DEFUNIAK, III      1993 $217,000    $ 75,950                        52,971.88       4,500               $ 11,122
 Senior Executive Vice    1992 $200,000    $ 62,000                        42,400.00       4,800               $ 11,240
 President, Birmingham    1991 $177,500    $ 78,100                       161,280.00       9,750
 Banking Group,
 AmSouth Bank N.A.
E. W. STEPHENSON, JR.     1993 $177,833    $ 71,133                        52,971.88       3,400               $  9,256
 Senior Executive Vice    1992 $150,000    $ 45,000                        27,825.00       3,150               $  8,560
 President, AmSouth       1991 $140,000    $ 52,488                       107,520.00       6,750
 and Chairman & Chief
 Executive Officer,
 AmSouth Bank of Florida
```

Label Handling

In [3]:
```
tags = [('name', None, (0, 1), None)]
tokens = tag(tokens, tags)
tokens = spread(tokens, 'name', 'name', edge=0)

print(reconstruct(tokens, 'all_tags'))
```

```
JOHN W. WOODS




C. STANLEY BAILEY


C. DOWD RITTER


WILLIAM A. POWELL, JR.




A. FOX DEFUNIAK, III




E. W. STEPHENSON, JR.
```

Figura 16 – Código Completo para Extrair os Dados do Campo "Nome"

Tokenization

```
In [2]: raw_file = open('stackexchange.txt', 'r')
        corpus = raw_file.read()

        print(corpus)

        tokens = tokenize(corpus)
```

```
                                    ANNUAL COMPENSATION                LONG-TERM COMPENSATION
                                ---------------------------------- ----------------------------------
                                                                            AWARDS            PAYOUTS
                                                                   ---------------------------- ----------
                                                                                SECURITIES
                                                          OTHER      RESTRICTED  UNDERLYING                 ALL OTHER
   NAME AND PRINCIPAL                                     ANNUAL       STOCK      OPTIONS/      LTIP       COPMPENSA-
        POSITION          YEAR SALARY ($)  BONUS ($) COMPENSATION ($) AWARD(S) ($)(1) SAR'S (#) PAYOUTS($) TION($)(3)
   ------------------     ---- ----------  --------- ---------------- -------------- ---------- ---------- ----------
JOHN W. WOODS             1993  $595,000    $327,250    There is no      $203,190.63     18,000                $ 29,295
 Chairman, President, &   1992  $545,000    $245,250    compensation      166,287.50     18,825    (2) Not     $ 29,123
 Chief Executive Officer  1991  $515,000    $283,251   required to be                    45,000    Applicable
 of AmSouth & AmSouth                                  disclosed in
 Bank N.A.                                             this column.
C. STANLEY BAILEY         1993  $266,667(4) $133,333                 117,012.50      4,500                $ 11,648
```

Figura 17 – Exemplo do "Stackexchange" – "Tokenização"

Label Handling

```
In [3]: tags = [('name', None, (0, 1), None), ('year_label', None, None, 'YEAR'), ('salary_label', None, None, 'SALARY'),
                ('bonus_label', None, None, 'BONUS'), ('awards_label', None, None, r'\bAWARD\b'),
                ('securities_label', None, None, 'SAR\'S'), ('all_compensations_label', None, None, 'TION')]
        tokens = tag(tokens, tags)

        for label, line, column, regex in tags[1:]:
            tokens = spread(tokens, label, label[:-6], mode='down', edge=0)
            tokens = spread(tokens, label[:-6], label[:-6], mode='down', edge=0, max_interval=10)
            tokens = retag(tokens, label, None)

        tokens = spread(tokens, 'name', 'name', edge=0)
        tokens = spread(tokens, 'name', 'position', mode='down')
        tokens = spread(tokens, 'position', 'position', edge=0)

        tags = [(None, None, None, '\-\-')]
        tokens = tag(tokens, tags)

        print(reconstruct(tokens, 'all_tags'))
```

```
JOHN W. WOODS            1993  $595,000    $327,250                $203,190.63     18,000              $ 29,295
 Chairman, President, &  1992  $545,000    $245,250                 166,287.50     18,825              $ 29,123
 Chief Executive Officer 1991  $515,000    $283,251                                45,000
 of AmSouth & AmSouth
 Bank N.A.
C. STANLEY BAILEY        1993  $266,667(4) $133,333                 117,012.50      4,500              $ 11,648
 Vice Chairman, AmSouth  1992  $210,000    $ 84,000                  42,400.00      4,800              $ 12,400
 & AmSouth Bank N.A.     1991  $186,750    $ 82,170                 161,280.00      9,750
C. DOWD RITTER           1993  $266,667(4) $133,333                 117,012.50      4,500              $ 13,566
 Vice Chairman, AmSouth  1992  $210,000    $ 84,000                  42,400.00      4,800              $ 12,920
 & AmSouth Bank N.A.     1991  $188,625    $ 82,995                 161,280.00      9,750
WILLIAM A. POWELL, JR.   1993  $211,335    $ 95,101                                11,000              $124,548
 President, AmSouth      1992  $330,000    $132,000                  98,050.00     11,100              $ 22,225
 and Vice Chairman,      1991  $308,000    $169,401                                24,000
 AmSouth Bank N.A.
 Retired in 1993
```

Figura 18 – Exemplo do "Stackexchange" – Definição de Campos

Segmentation and Reading Order

```
In [4]: tokens = chop(tokens, 'name')
        tokens = chop(tokens, 'year', mode='vertical')
        tokens = chop(tokens, 'year')

        for section in tokens:
            print(reconstruct(section, 'all_tags'))
            print('*' * 120)
```

```
************************************************************************************************************************
JOHN W. WOODS
 Chairman, President, &
 Chief Executive Officer
 of AmSouth & AmSouth
 Bank N.A.
************************************************************************************************************************
1993  $595,000    $327,250                $203,190.63     18,000              $ 29,295
************************************************************************************************************************
1992  $545,000    $245,250                 166,287.50     18,825              $ 29,123
************************************************************************************************************************
1991  $515,000    $283,251                                45,000
************************************************************************************************************************
C. STANLEY BAILEY
 Vice Chairman, AmSouth
 & AmSouth Bank N.A.
************************************************************************************************************************
```

Figura 19 – Exemplo "Stackexchange" – Segmentação e Ordem de Leitura

Data Extraction and Definition of Hierarchy

```
In [5]: optional_fields=[('awards', 'bonus'), ('all_compensations', 'securities')]
last_section = ['year', 'salary', 'bonus', 'securities']
token_hierarchy = extract(tokens, last_section=last_section, optional_fields=optional_fields)

old_hierarchy = token_hierarchy
token_hierarchy = []
for entry in old_hierarchy:
    entry['position'] = entry['position'].replace('\n', ' ').replace('\r', '')
    token_hierarchy.append(entry)

display(token_hierarchy)
```

```
[{'all_compensations': '$ 29,295',
  'awards': '$203,190.63',
  'bonus': '$327,250',
  'name': 'JOHN W. WOODS',
  'position': 'Chairman, President, & Chief Executive Officer of AmSouth & AmSouth Bank N.A.',
  'salary': '$595,000',
  'securities': '18,000',
  'year': '1993'},
 {'all_compensations': '$ 29,123',
  'awards': '166,287.50',
  'bonus': '$245,250',
  'name': 'JOHN W. WOODS',
  'position': 'Chairman, President, & Chief Executive Officer of AmSouth & AmSouth Bank N.A.',
  'salary': '$545,000',
  'securities': '18,825',
```

Figura 20 – Exemplo "Stackexchange" – Extração dos dados e Definição de Hierarquia

Saving Data

```
In [6]: result = save(token_hierarchy)

print(result)
```

```
Stored in: C:\Users\equidna\Dropbox\python\PositionParser\output.csv
```

Figura 21 – Exemplo "Stackexchange" – Guardar os dados

Outro teste de extração foi feito a um ficheiro de texto extraído de um PDF com resultados de atletismo como pode ser observado nas figuras 22, 23, 24 e 25.

Tokenization

```
In [60]: raw_file = open('athletics.txt', 'r')
corpus = raw_file.read()
corpus = strip_blanks(corpus)

print(corpus)

tokens = tokenize(corpus)
```

```
                        Associação de Atletismo de Viana do Castelo                                    1749016
                                              Comunicado de Resultados
                                    Quilometro Jovem e Salto em Altura Distrital
                                            Centro de Estágios de Melgaço
                                                        Melgaço

Jornada de: sábado, 25 de Março de 2017
      100 Metros planos                    (Fem)       1 500 Metros                                (Fem)
      Prova Extra 1                Vento:        0,0  Prova Extra 1
1ª Daniela Cruz          01 Juv  CAOV     1 4,14       2ª Mariana Sousa          99 Jun    CYCL    5 2 6,18
2ª Leticia Costa         98 Jun  CYCL     1 4,21       3ª Joana Rocha            98 Jun    CAAV    6 3 3,13
3ª Mickaela Costa        01 Juv  CAAV     1 4,22
4ª Ines Lage             02 Ini  CYCL     1 4,31
                                                       Salto em Altura                             (Fem)
      Salto em Altura                    (Masc)                        Final 1
                Final 1                                 1ª Marta Lisboeta         04 Inf    CAOV    1,38
                                                        2ª Carina Nogueira        04 Inf    OGADD   1,35
```

Figura 22 – "Tokenização" dos Dados de Atletismo

Label Handling

In [61]:
```
line_tags = [('data', None, None, 'Jornada de')]
tag_lines(tokens, line_tags)

tags = [('delim_pagina', None, None, '\x0c'), ('prova', (3, 3), None, None), ('local', (4, 4), None, None),
        (None, None, None, 'Jornada'), (None, None, None, 'de:'), ('sexo', None, None, '.*Fem.*|.*Masc.*'),
        ('posicao', None, None, u'\w+(?:ª|º)'), ('serie', None, None, 'Serie'), ('serie', None, None, 'Prova'),
        ('serie', None, None, 'Final'), ('classific_clubes', None, None, 'classific')]
tokens = tag(tokens, tags)

tokens = spread(tokens, 'serie', 'serie', edge=0)
tokens = mirror(tokens, 'sexo', 'modalidade', -1)
tokens = spread(tokens, 'modalidade', 'modalidade', 'left', edge=0)
tokens = spread(tokens, 'classific_clubes', 'classific_clubes', edge=0, mode='left', replace_enabled=True)
tokens = retag(tokens, 'classific_clubes', None)
tokens = mirror(tokens, 'posicao', 'atleta')
tokens = spread(tokens, 'atleta', 'atleta', edge=0)
tokens = mirror(tokens, 'atleta', 'nascimento')
tokens = mirror(tokens, 'nascimento', 'escalao')
tokens = mirror(tokens, 'escalao', 'clube')
tokens = mirror(tokens, 'clube', 'marca')
tokens = spread(tokens, 'marca', 'marca', edge=0)

print(reconstruct(tokens, 'all_tags'))
```

```
                                    Quilometro Jovem e Salto em Altura Distrital
                                            Centro de Estágios de Melgaço

            sábado, 25 de Março de 2017
      100 Metros planos                    (Fem)        1 500 Metros                                (Fem)
      Prova Extra 1                                     Prova Extra 1
1ª Daniela Cruz          01 Juv  CAOV     1 4,14       2ª Mariana Sousa          99 Jun    CYCL    5 2 6,18
2ª Leticia Costa         98 Jun  CYCL     1 4,21       3ª Joana Rocha            98 Jun    CAAV    6 3 3,13
3ª Mickaela Costa        01 Juv  CAAV     1 4,22
4ª Ines Lage             02 Ini  CYCL     1 4,31
                                                       Salto em Altura                             (Fem)
      Salto em Altura                    (Masc)                        Final 1
                Final 1                                 1ª Marta Lisboeta         04 Inf    CAOV    1,38
                                                        2ª Carina Nogueira        04 Inf    OGADD   1,35
1º Valdemar Dantas       05 Inf  ADCL     1,31         3ª Lara Barros            04 Inf    OGADD   1,26
2º Rodrigo Ferreira      04 Inf  CAM-VC   1,31         4ª Sofia Manso            04 Inf    OGADD   1,20
```

Figura 23 – Definição de Campos dos Dados de Atletismo

Segmentation and Reading Order

In [62]:
```
paginas = chop(tokens, 'delim_pagina')
zonas = chop(paginas, 'data', displacement=1)
colunas = chop(zonas, 'posicao', 'vertical', [None, None, 15, None], -1, 2)
modalidades = chop(colunas, 'modalidade')
series = chop(modalidades, 'serie')
posicoes = chop(series, 'posicao')

for posicao in posicoes:
    print(reconstruct(posicao))
    print('----------------------------------------------------------------------------------------------------------------------------------------------------------------')
```

```
                        Associação de Atletismo de Viana do Castelo                                                 1749016
                                                Comunicado de Resultados
                                        Quilometro Jovem e Salto em Altura Distrital
                                                Centro de Estágios de Melgaço
                                                        Melgaço
Jornada de: sábado, 25 de Março de 2017
----------------------------------------------------------------------------------------------------------------------------------------------------------------
100 Metros planos                    (Fem)
----------------------------------------------------------------------------------------------------------------------------------------------------------------
Prova Extra 1                Vento:          0,0
----------------------------------------------------------------------------------------------------------------------------------------------------------------
1ª Daniela Cruz           01 Juv  CAOV     1 4,14
----------------------------------------------------------------------------------------------------------------------------------------------------------------
2ª Leticia Costa          98 Jun  CYCL     1 4,21
----------------------------------------------------------------------------------------------------------------------------------------------------------------
3ª Mickaela Costa         01 Juv  CAAV     1 4,22
----------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Figura 24 – Segmentação e Ordem de Leitura dos Dados de Atletismo

Data Extraction and Definition of Hierarchy

```
In [63]: old_posicoes = posicoes
         posicoes = []
         for posicao in old_posicoes:
             new_segment = retag(posicao, 'delim_pagina', None)
             posicoes.append(new_segment)

         last_section = ['posicao', 'atleta', 'nascimento', 'escalao', 'clube', 'marca']
         token_hierarchy = extract(posicoes, last_section=last_section, optional_fields=[('serie', 'sexo')])

         display(token_hierarchy)
```

```
[{'atleta': 'Daniela Cruz',
  'clube': 'CAOV',
  'data': 'sábado, 25 de Março de 2017',
  'escalao': 'Juv',
  'local': 'Centro de Estágios de Melgaço',
  'marca': '1 4,14',
  'modalidade': '100 Metros planos',
  'nascimento': '01',
  'posicao': '1ª',
  'prova': 'Quilometro Jovem e Salto em Altura Distrital',
  'serie': 'Prova Extra 1',
  'sexo': '(Fem)'},
 {'atleta': 'Leticia Costa',
  'clube': 'CYCL',
  'data': 'sábado. 25 de Marco de 2017'.
```

Figura 25 – Extração dos dados e Definição de Ordem de Leitura

4.5 Terceira Fase – Estimativa da Relevância do *PositionParser*

Vendo o protótipo de uma perspetiva de potencial valor monetário efetuou-se uma pesquisa ao website de *freelancing* "ODesk" (https://www.upwork.com), em que é possível observar a existência de uma grande quantidade de oferta de trabalho relacionado com "*data entry*". Tomando como exemplo o caso apresentado nas figuras 26 e 27 onde é oferecida a quantia de 13.11 dólares por hora, por um trabalho com uma duração entre 10 a 30 horas semanais. Considerando que o trabalho tem uma duração de 10 horas semanais e o cliente apenas está disposto a pagar 13 dólares:

10 x 13 x 4 = 520 dólares mensais

Podemos calcular que o lucro seria cerca de 520 dólares por mês.

Figura 26 – Caso Real de Trabalho de Introdução de Dados

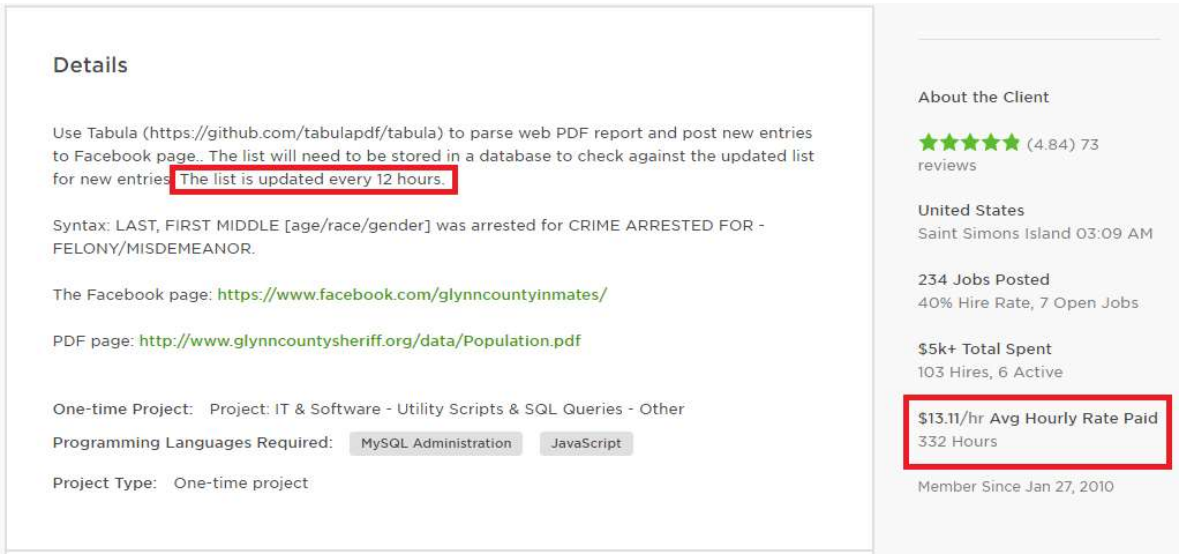
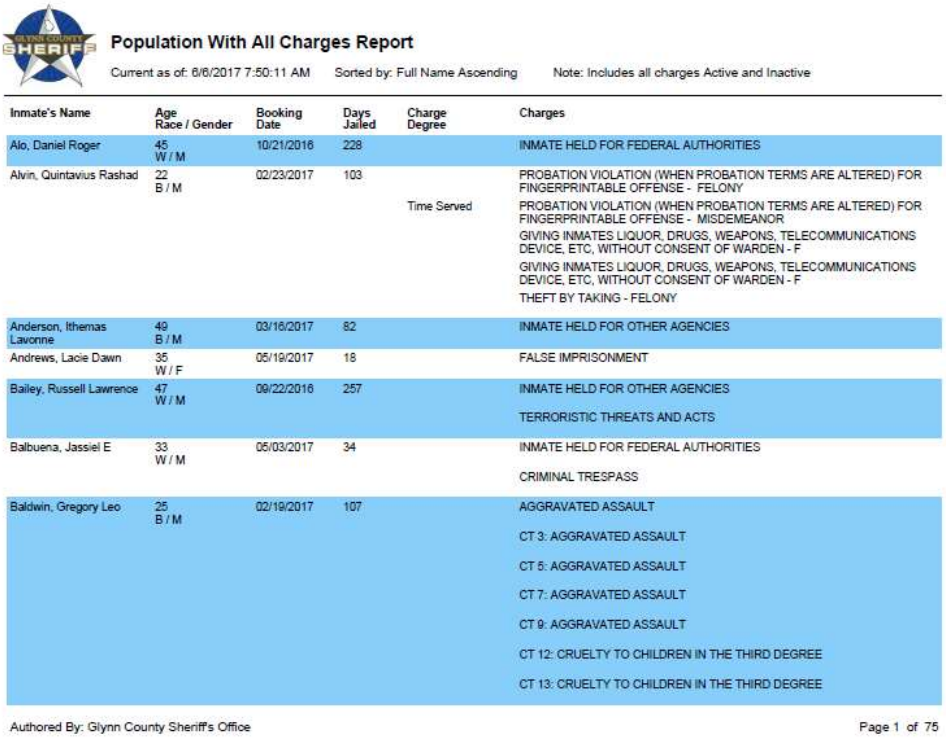


Figura 27 – Caso Real de Oferta de 13.11 Dólares por Cada Hora de Trabalho

Observando com maior atenção a figura 27 verifica-se que é um ficheiro PDF atualizado de 12 em 12 horas, logo é necessário analisar 60 ficheiros por mês o que significa um rendimento de cerca de 8.5 dólares por ficheiro:

520 / (2 x 30) = 8.667 dólares por ficheiro

Observando o exemplo dos ficheiros a serem analisados representado na figura 28, é possível concluir que é possível efetuar uma extração semelhante à representada nos testes anteriores.



Population With All Charges Report

Current as of: 6/6/2017 7:50:11 AM Sorted by: Full Name Ascending Note: Includes all charges Active and Inactive

Inmate's Name	Age Race / Gender	Booking Date	Days Jailed	Charge Degree	Charges
Alo, Daniel Roger	45 W / M	10/21/2016	228		INMATE HELD FOR FEDERAL AUTHORITIES
Alvin, Quintavius Rashad	22 B / M	02/23/2017	103		PROBATION VIOLATION (WHEN PROBATION TERMS ARE ALTERED) FOR FINGERPRINTABLE OFFENSE - FELONY
				Time Served	PROBATION VIOLATION (WHEN PROBATION TERMS ARE ALTERED) FOR FINGERPRINTABLE OFFENSE - MISDEMEANOR
					GIVING INMATES LIQUOR, DRUGS, WEAPONS, TELECOMMUNICATIONS DEVICE, ETC, WITHOUT CONSENT OF WARDEN - F
					GIVING INMATES LIQUOR, DRUGS, WEAPONS, TELECOMMUNICATIONS DEVICE, ETC, WITHOUT CONSENT OF WARDEN - F
					THEFT BY TAKING - FELONY
Anderson, Ithemas Lavonne	49 B / M	03/16/2017	82		INMATE HELD FOR OTHER AGENCIES
Andrews, Lacie Dawn	35 W / F	05/19/2017	18		FALSE IMPRISONMENT
Bailey, Russell Lawrence	47 W / M	09/22/2016	257		INMATE HELD FOR OTHER AGENCIES
					TERRORISTIC THREATS AND ACTS
Balbuena, Jassiel E	33 W / M	05/03/2017	34		INMATE HELD FOR FEDERAL AUTHORITIES
					CRIMINAL TRESPASS
Baldwin, Gregory Leo	25 B / M	02/19/2017	107		AGGRAVATED ASSAULT
					CT 3: AGGRAVATED ASSAULT
					CT 5: AGGRAVATED ASSAULT
					CT 7: AGGRAVATED ASSAULT
					CT 9: AGGRAVATED ASSAULT
					CT 12: CRUELTY TO CHILDREN IN THE THIRD DEGREE
					CT 13: CRUELTY TO CHILDREN IN THE THIRD DEGREE

Authored By: Glynn County Sheriff's Office Page 1 of 75

Figura 28 – Exemplo do Ficheiro a ser Retirado os Dados de Uma Oferta Real de Trabalho

Como as tabelas podem ter uma variedade de formatos, a sua extração de dados é uma área extremamente difícil e extensível. No entanto, também é uma área de grande importância pois é nas tabelas onde provavelmente se encontra uma "maior densidade de informação" (Martha Perez-Arriaga et al., 2016).

A solução criada é potencialmente mais adequada, face às existentes, pois em vez de ser aplicado automaticamente o mesmo algoritmo, o trabalho de extração de dados é feito à medida de tabelas com a mesma estrutura e é executado por alguém consciente das pequenas nuances das mesmas.

O *PositionParser*, no seu estado atual, é apenas um protótipo e só poderá ser usada por um programador ou um analista de dados.

O objetivo é inicialmente dividir o texto em fragmentos semelhantes chamados de "tokens" para trazer consistência ao processo de análise. A segmentação de texto pode melhorar significativamente a performance de vários algoritmos de mineração de texto, separando documentos heterogéneos em fragmentos homogéneos e assim facilitando o processamento subsequente (Dadachev et al., 2014).

Através deste exemplo é possível ter uma referência dos custos que o projeto de extração (de maior dimensão) teria, caso a ferramenta *PositionParser* não fosse implementada.

4.6 Terceira Fase – Extração dos Dados e Documentação

Com a ajuda do *PositionParser*, é criado um algoritmo para extrair a informação, diferente para cada conjunto de ficheiros de igual formato.

As funções do *PositionParser* são usadas de acordo com o respetivo segmento do fluxo de trabalho, como demonstrado na figura 29.

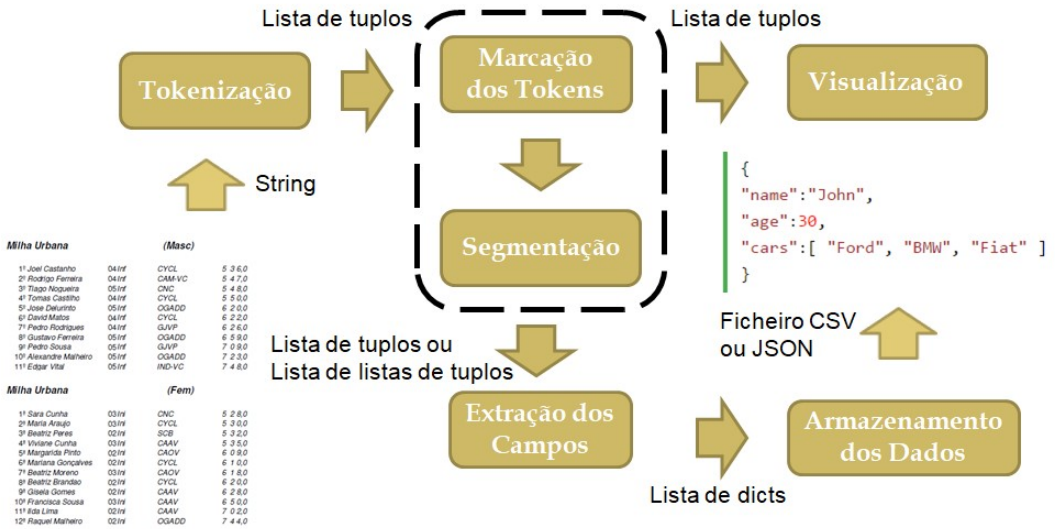


Figura 29 – Fluxo de Trabalho do Protótipo *PositionParser*

As funções da ferramenta estão divididas da seguinte forma:

- "Tokenização" do Texto: strip_blanks, tokenize;
- Marcação dos "tokens": tag, retag, tag_lines, mirror, spread;
- Visualização: reconstruct;
- Segmentação: chop, remove_untagged, merge;
- Hierarquia e extração de campos: extract;
- Armazenamento dos dados: save.

A função "strip_blanks" é usada para remover linhas em branco do texto original. O número de linhas em branco pode variar entre textos, mesmo quando a estrutura é idêntica. Sendo assim, isto é mais um passo para normalizar o texto. Esta função retorna o *corpus* sem as linhas em branco e tem 2 argumentos:

1. corpus (string) – a string de texto original;
2. line_delimiter (string) – o caractere que define uma nova linha, que por defeito é "\n".

A função "tokenize" é usada para converter o texto em "tokens", convertendo a *string* do *corpus* numa lista de tuplos. Cada tuplo com 4 valores: uma *string* representativa duma palavra, um valor inteiro representativo da linha do início da palavra, outro da coluna e uma *string* representativa do rótulo que dá significado à palavra. Precisa de 3 argumentos:

1. corpus (string) – texto original;
2. line_delimiter (string) – caractere que define uma mudança de linha. Por defeito o valor é "\n";
3. token_delimiter (string) – caractere representativo do espaço entre palavras. Por defeito é um espaço em branco.

A função "tag" marca os "tokens" de uma lista e leva 2 argumentos:

1. tokens (lista de tuplos) – lista de "tokens" com a seguinte estrutura: [(valor, linha de início, coluna de início, rótulo), …];
2. tags (lista de tuplos) – lista de indicação de marcações, usada para rotular qualquer "token" entre os intervalos definidos e que sigam a validação "regex" definida, com a seguinte estrutura: [(classificador, (intervalo mínimo de linha, intervalo máximo de linha), (intervalo mínimo de coluna, intervalo mínimo de coluna), validação "regex"), …]. Caso não exista uma validação regex ou intervalo de linha ou coluna para a rotulagem a constante "None" deve ser usada da seguinte forma: [('classificador', None, None, None)], este argumento rotula todos os "tokens" existentes com a etiqueta "classificador".

A função "retag" simplesmente altera o rótulo dos "tokens" e possui 3 argumentos:

1. tokens (lista de tuplos) – lista de "tokens";

2. original_tag (string) – o rótulo original a ser alterado;
3. new_tag (string) – o novo rótulo.

Para marcar todos os "tokens" de uma linha a função "tag_lines" deve ser usada. Possui 3 argumentos:

1. tokens (lista de tuplos) – lista de "tokens";
2. line_tags (lista de tuplos) – uma lista de indicação de marcações, usada para rotular qualquer "token" entre os intervalos definidos e que sigam a validação "regex" definida, com a seguinte estrutura: [(classificador, (intervalo mínimo de linha, intervalo máximo de linha), (intervalo mínimo de coluna, intervalo mínimo de coluna), validação "regex"), …]. Caso não exista uma validação regex ou intervalo de linha ou coluna para a rotulagem a constante "None" deve ser usada da seguinte forma: [('classificador', None, None, None)];
1. space_break (string) – o caractere representativo do espaço entre "tokens". Por defeito é um espaço em branco.

Esta função é utilizada para evitar o uso da função "tag" seguido de "spread" para marcar uma linha.

A função "mirror" marca "tokens" baseando-se na posição de etiquetas já existentes, da maneira observada na figura 30. Possui 7 argumentos, 3 obrigatórios e 4 opcionais:

1. tokens (lista de tuplos) – lista de tokens;
2. original_tag (string) – a etiqueta original que servirá de referência;
3. destination_tag (string) – a etiqueta de marcação dos "tokens";
4. steps (inteiro) – o número de "tokens" entre os que possuem uma etiqueta original e os que irão ser marcados. Por defeito o número de passos é 1;
5. mode (string) – define se a contagem é feita na horizontal ou vertical. Os valores aceites são "horizontal" ou "vertical" e por defeito é "horizontal";
6. condition (string ou "None") – Uma condição regex que, quando correspondida, permite que o "token" seja marcado. Caso não seja necessária uma condição, a variável "None" deve ser usada. Por defeito o valor da condição é "None";
7. replace_enabled (booleano) – Aceita 2 valores: "True" ou "False". Quando ativado permite que um "token" seja marcado mesmo no caso de já ter uma etiqueta. Por defeito esta variável encontra-se desativada ("False").

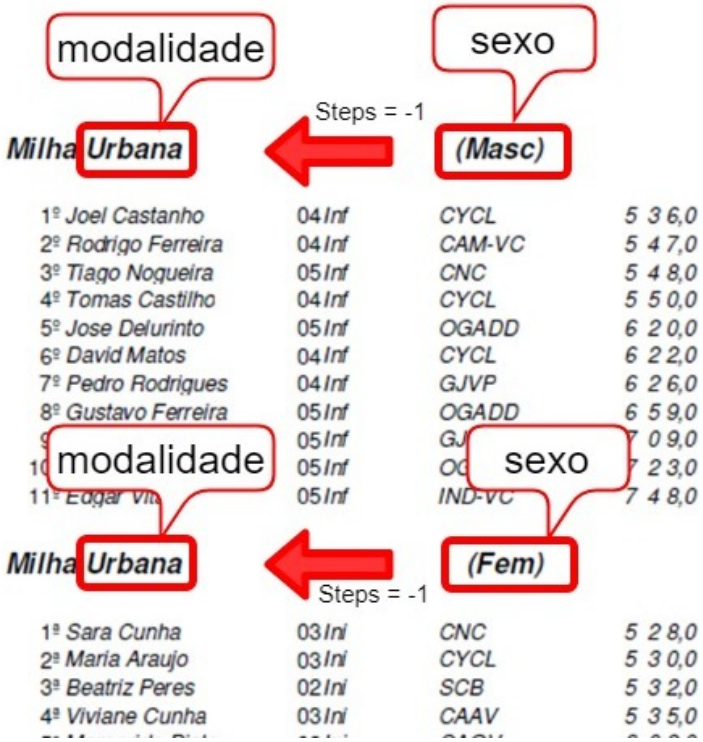


Figura 30 – Representação Visual da Função "mirror"

A função "spread" marca todos os "tokens" em qualquer direção. Pode levar até 9 argumentos, 3 mandatórios e 6 opcionais:

1. tokens (lista de tuplos) – lista de tokens;
2. start_tag (string) – a etiqueta original que servirá de referência;
3. new_tag (string) – a etiqueta de marcação dos "tokens";
4. mode (string) – existem 4 modos de marcação correspondentes às 4 direções que a marcação pode tomar: "right", "left", "up" e "down". Por defeito a marcação assumida é "right";
5. max_interval (inteiro) – é o intervalo máximo entre cada "token" a ser marcado, se o intervalo máximo for atingido a marcação pára. Por defeito o intervalo máximo é 1;
6. stop_regex (string ou "None") – Uma condição regex que, quando correspondida, pára a marcação de "tokens". Se nenhuma marcação é necessária a variável "None" deve ser usada. Por defeito o seu valor é "None";
7. stop_tags (lista de strings ou lista vazia) – uma lista de etiquetas, que quando correspondidas, pára com a marcação. Por defeito a variável é uma lista vazia;
8. edge (inteiro) – este argumento permite aumentar ou diminuir os limites dos tokens iniciais. Por defeito o limite é 1;
9. replace_enabled (booleano) – Aceita 2 valores: "True" ou "False". Quando ativado permite que um "token" seja marcado mesmo no caso de já ter uma etiqueta. Por defeito esta variável encontra-se desativada ("False").

Uma exemplificação do seu funcionamento está exposta na figura 31.

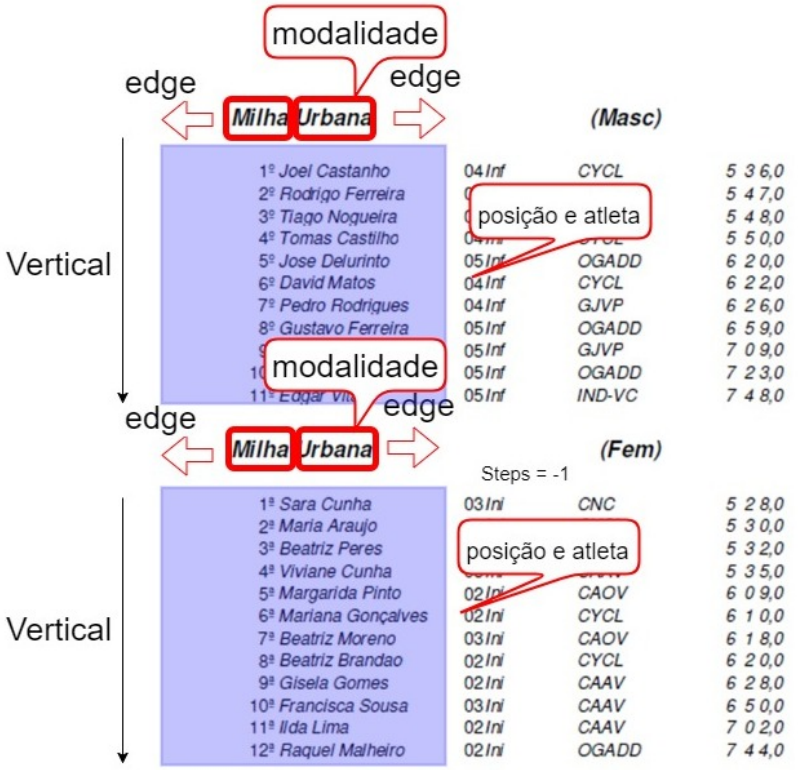


Figura 31 – Representação Visual da Função "spread"

Utilizada para visualizar o trabalho a ser feito a função "reconstruct" permite reconverter a lista de tokens (tuplos) novamente numa string possibilitando ao utilizador visualizar quais os "tokens" marcados com quais etiquetas. Possui 4 argumentos, sendo 3 deles opcionais:

1. tokens (lista de tuplos) – lista de "tokens";
2. tag (string ou "None") – o nome da etiqueta dos "tokens" a serem mostrado. Caso seja necessário ver todos os tokens etiquetado a *string* "all_tags" deve ser usada. Para ver todos os "tokens" não marcados a variável "None" deve ser usada. Para observar todos os "tokens" (marcados e não marcados) a *string* "all" deve ser introduzida. Por defeito o valor é "all";
3. line_break (string) – o caractere que define uma nova linha. Por defeito o valor é "\n";
4. space_break (string) – o caractere representativo do espaço entre "tokens". Por defeito é um espaço em branco.

A função "chop" define a ordem de leitura da informação, isto é feito através da divisão de listas de "tokens" em listas de listas, onde cada lista representa uma secção e cada secção representa um grupo de "tokens" que, por sua vez, são representados por tuplos. Possui 6 argumentos (4 deles opcionais):

1. token_order (lista de tuplos ou lista de lista de tuplos se o texto já estiver seccionado e uma nova divisão for necessária) – lista ou lista de listas de tokens;

2. tag (string) – a etiqueta dos “tokens” que servirão de delimitador de cada segmento;
3. mode (string) – o tipo de segmentação do texto pode ser horizontal ou vertical, para esse efeito existem 2 modos “horizontal” e “vertical”. Por defeito o modo “horizontal” está ativo;
4. área (lista de 4 inteiros) – Uma divisão pode ser aplicada apenas numa só área. Se nenhuma área for selecionada “None” deve ser introduzido. Por defeito “None” é selecionado. A área a segmentar é definida da seguinte forma: [linha mínima, linha máxima, coluna mínima, coluna máxima];
5. displacement (inteiro) – os valores dos pontos de segmentação podem ser alterado com este argumento. Por defeito não é aplicada qualquer alteração, o valor é 0;
6. min_interval (inteiro positivo) – é a distância mínima entre um delimitador e o delimitador precedente para que este seja aceite. Por defeito o intervalo é 1.

A função “remove_untagged” remove todos os “tokens” que até então não foram marcados. Possui apenas um argumento obrigatório:
1. tokens (lista de tuplos) – lista de “tokens”.

A função “merge” junta “tokens” num só tuplo e possui 4 argumentos (3 deles opcionais):
1. token_order (lista de lista de tuplos) – lista ou lista de listas de tokens;
2. mode (string) – a maneira como os tokens serão agregados. Se “line” for escolhido apenas serão agrupados os “tokens” com a mesma etiqueta e na mesma linha de cada segmento, caso o modo “segment” seja escolhido serão agrupados os “tokens” com a mesma etiqueta e no mesmo segmento;
3. line_break (string) – o caractere que define uma nova linha. Por defeito o valor é “\n”;
4. space_break (string) – o caractere representativo do espaço entre “tokens”. Por defeito é um espaço em branco.

A função “extract” define as hierarquias as hierarquias das marcações e extrai os valores dos “tokens”. Retorna uma lista de dicts; cada dicionário representa uma entrada, com uma chave para cada etiqueta e um valor para cada valor do “token” correspondente. Possui 5 argumentos, 2 obrigatórios e 3 opcionais:
1. token_order (lista de listas de tuplos) – lista de listas de tokens com a ordem já definida;
2. end_field (lista de strings) – representa uma lista de todos os campos (etiquetas) de hierarquia mais baixa;
3. optional_fields (lista de tuplos) – uma lista de campos opcionais compostos por 2 valores: o primeiro é uma *string* que mostra o valor do campo opcional e o segundo é uma *string* que representa o campo assistente que irá fazer *reset* ao

valor do campo opcional. Geralmente o campo assistente é o campo imediatamente antes do campo opcional na ordem de leitura;

4. line_break (string) – o caractere que define uma nova linha. Por defeito o valor é "\n";
5. space_break (string) – o caractere representativo do espaço entre "tokens". Por defeito é um espaço em branco.

A função "save" transforma e armazena os dados formatados num ficheiro CSV ou JSON. Possui 3 argumentos e 2 deles são opcionais:

1. field_values (lista de dicts) – uma lista das entradas devolvida pela função "extract" a serem armazenadas;
2. file_dir (string com uma diretoria válida) – uma string com o caminho e nome do ficheiro de destino. Por defeito a função armazena os dados num ficheiro de nome "output.csv" na diretoria de trabalho;
3. file_format (string) – o argumento que define o formato do ficheiro. Pode ser "csv" ou "json". Por defeito o valor é "csv".

Com as funções deste módulo é possível criar algoritmos de extração compatíveis com uma grande quantidade de ficheiros; um exemplo seria:

```
def pdf_parser(diretoria_entrada, diretoria_saida):
    # escolha dos ficheiros
    ficheiro_simples = open(diretoria_entrada, 'r')
    corpus = ficheiro_simples.read()
    ficheiro_simples.close()
    if 'Prova:' not in corpus:
        return None
    else:
        pass

    # tokenização
    corpus = strip_blanks(corpus)
    tokens = tokenize(corpus)
    corpus = None

    # lidar com etiquetas
    tags = [('ancora_prova', None, None, 'Prova:'),
            ('ancora_data', None, None, 'Data:'),
            ('sexo', None, None,
             '(?i)masc.*|fem.*|(^(M|F)[0-9][05]$)'),
            ('ancora_classificacao', None, None, 'Class'),
            ('ancora_marca', None, None, 'Tempo')]
    tokens = tag(tokens, tags, replace_enabled=False)
    tokens = spread(tokens, 'ancora_prova', 'prova', mode='down',
                    stop_regex='Organização:')
    tokens = retag(tokens, 'ancora_prova', None)
    tokens = spread(tokens, 'prova', 'prova', max_interval=5)
    tokens = spread(tokens, 'prova', 'local', max_interval=100)
    tokens = spread(tokens, 'ancora_data', 'data', mode='down',
                    stop_regex='Pontos')
    tokens = retag(tokens, 'ancora_data', None)
```

```
    tokens = spread(tokens, 'data', 'data', max_interval=10,
                    mode='left')
    tokens = spread(tokens, 'data', 'data', max_interval=10)
    tokens = mirror(tokens, 'sexo', 'escalao', steps=-1)
    tokens = mirror(tokens, 'sexo', 'modalidade')
    tokens = spread(tokens, 'ancora_classificacao', 'classificacao',
                    mode='down')
    tokens = retag(tokens, 'ancora_classificacao', None)
    tokens = mirror(tokens, 'classificacao', 'atleta', steps=2)
    tokens = spread(tokens, 'atleta', 'atleta')
    tokens = spread(tokens, 'ancora_marca', 'marca', mode='down',
                    stop_regex='[^0-9,\.]')
    tokens = retag(tokens, 'ancora_marca', None)
    tokens = mirror(tokens, 'marca', 'clube', steps=-1)
    tokens = spread(tokens, 'clube', 'clube', mode='left')
    #print(reconstruct(tokens[:], 'all tags'))
    #print(diretoria_entrada)

    # segmentação e ordem de leitura
    segmentos = chop(tokens, 'sexo')
    tokens = None
    #for segmento in segmentos:
    #    print(reconstruct(segmento, 'all_tags'))
    #    print('-----------------------------------------------------')

    # fusão de tokens
    # remover tokens sem tag para a extração de campos
    segmentos = [remove_untagged(s) for s in segmentos]
    segmentos = merge(segmentos, mode='line')

    # extração de dados
    # campos de hierarquia mais baixa
    ultimos_campos = ['classificacao', 'atleta', 'clube', 'marca']
    # campos opcionais que não pertençam à última secção
    campos_opcionais = []
    campos = extract(segmentos, ultimos_campos, campos_opcionais)
    del segmentos[:]

    # guardar dados em ficheiro CSV
    resultado = save(campos, diretoria_saida)
    campos.clear()
    print(resultado)
```

Esta função pode ser aplicada em todos os ficheiros do distrito dos açores cujo formato tem “Prova” no texto.

4.7 Terceira Fase – Comparação Com as Outras Ferramentas de Extração

Na tabela 8 é possível observar como a ferramenta criada se compara com as restantes já existentes.

	Tabula	PDFTables	PDFix	PositionParser (Protótipo)
Identificação dos Campos	Não	Não	Não	Manual
Ordem de Leitura e Deteção de Tabelas	Manual Auto	Auto	Auto	Manual
Hierarquia dos Campos	Não	Não	Auto	Manual
Extração para um Formato Estruturado	Sim	Sim	Sim	Sim
Escalabilidade (API)	Não	Sim	Sim	Sim
Ficheiros de Origem	PDF	PDF	PDF	TXT
Interface Gráfica	Sim	Sim	Não	Não
Preço	Gratuito	De 0.02$ a 0.03$cadapágina	De 490$ a 2270$	Gratuito

Tabela 8 – Comparação Entre Ferramentas de Extração de Dados e o *PositionParser*

4.8 Quarta Fase – Uniformização e Limpeza dos Dados

Tendo a informação estruturada em ficheiros CSV é feito o *data Wrangling* utilizando a ferramenta "OpenRefine".

Uma vez que não é possível trabalhar com uma quantidade maior do que 1 milhão de linhas de uma só vez, é necessário trabalhar os ficheiros distrito a distrito, para isso basta criar um projeto clicando em "Create Project" e "Escolher Ficheiros" como demonstrado na figura 32.

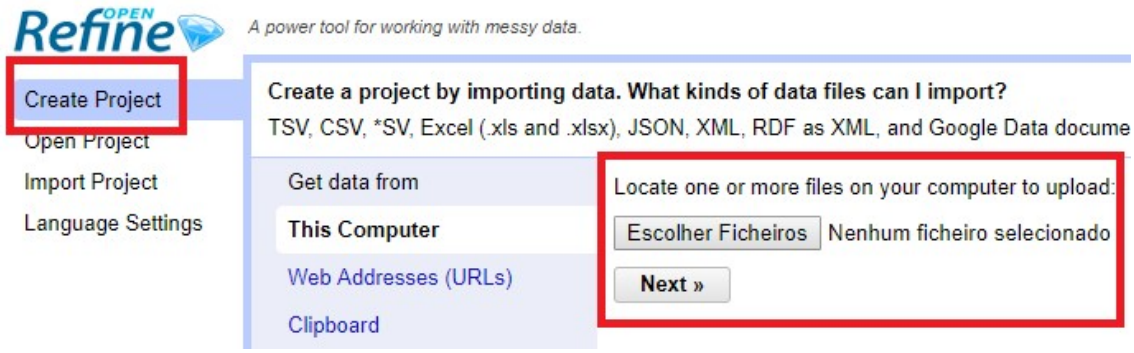


Figura 32 – Criação de um Novo Projeto com OpenRefine

Mais à frente é possível selecionar a codificação dos ficheiros, o nome a dar ao projeto e o separador dos valores, como demonstrado na figura 33.

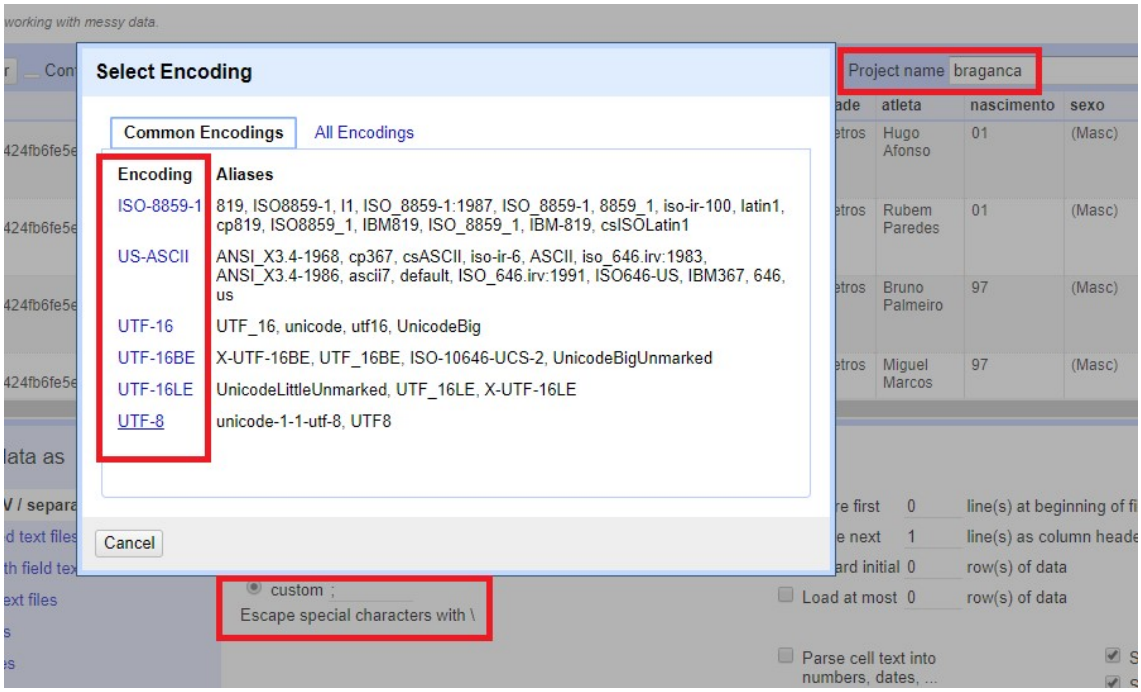


Figura 33 – Opções de Análise dos Ficheiros CSV

Para resumir todos os valores diferentes de um campo, basta clicar na seta do campo que queremos facetar e escolher “facet” e “text facet” demonstrado na figura 34.

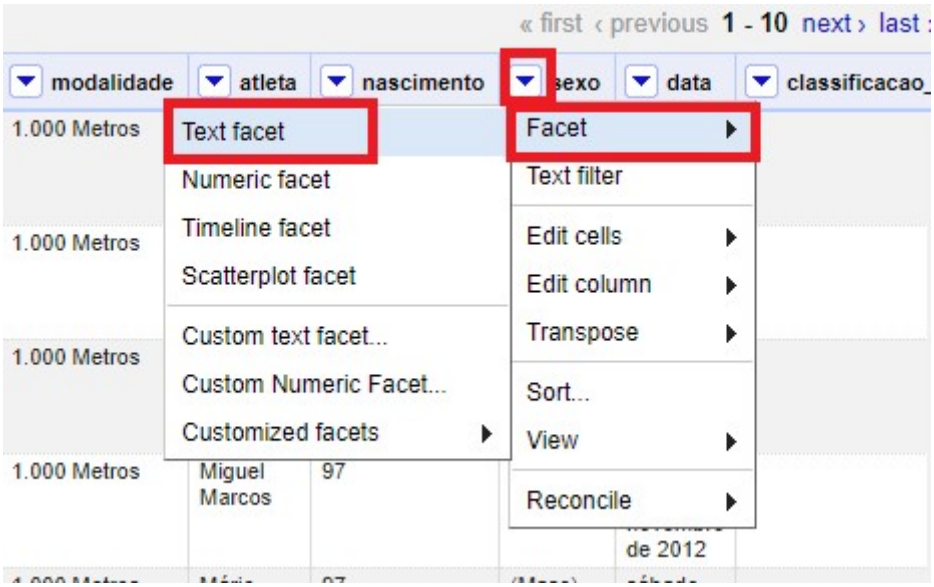


Figura 34 – Facetar valores de um Campo no OpenRefine

Isto criará uma janela com todos os valores e a sua respetiva quantidade semelhante à apresentada na figura 35. As linhas a apresentar podem ser selecionadas por cada valor único e múltiplas facetas podem ser criadas.

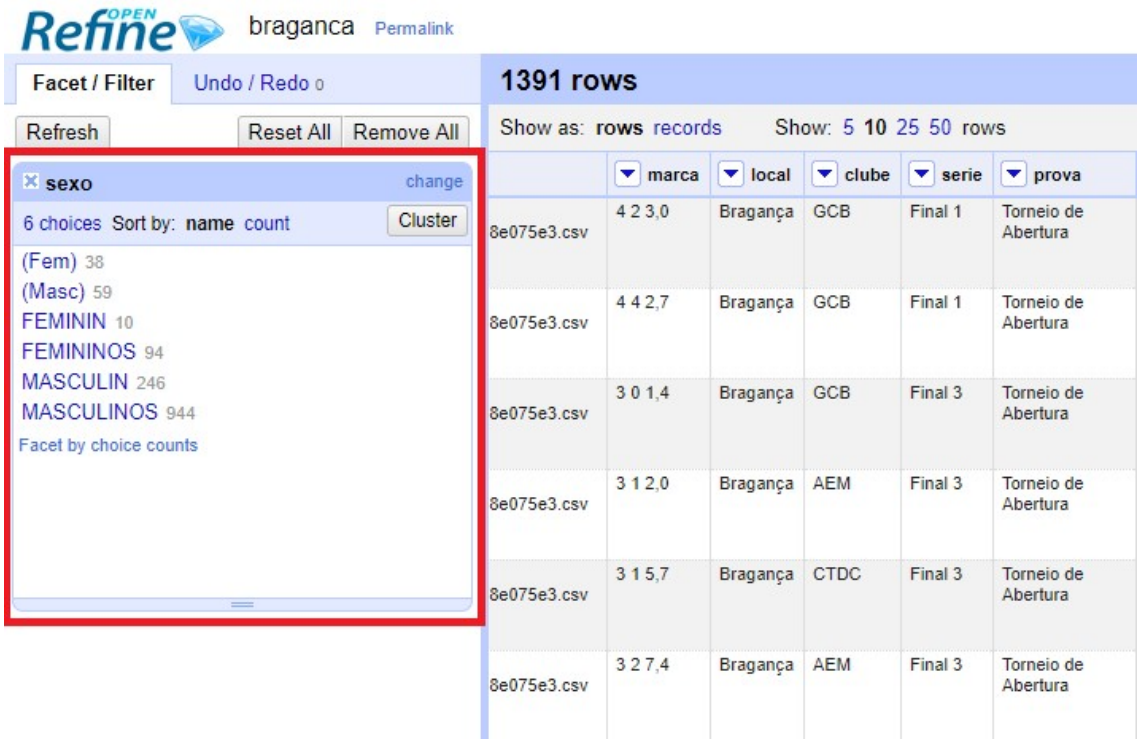


Figura 35 – Valores Únicos de Uma das Colunas

É possível editar todos os valores de um dos nomes da faceta passando com o cursor por cima do nome a alterar, clicando em "edit" e alterar para o nome pretendido, como demonstrado na figura 36.

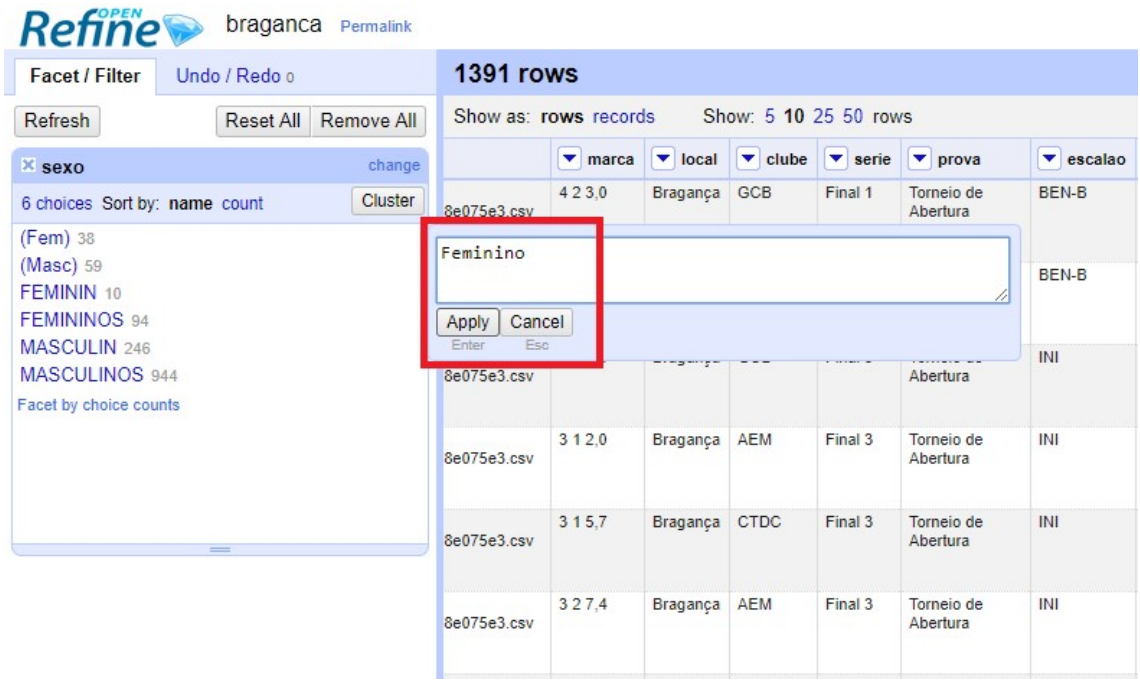


Figura 36 – Editar um dos Nomes De uma Faceta

Para efetuar este processo de forma semiautomática é necessário clicar em "Cluster" por cima da janela de faceta (exemplificado na figura 37), e escolher o algoritmo a aplicar para tentar encontrar valores de significado igual (ver figura 38).

Figura 37 – Botão Para Aceder ao *Clustering* no OpenRefine

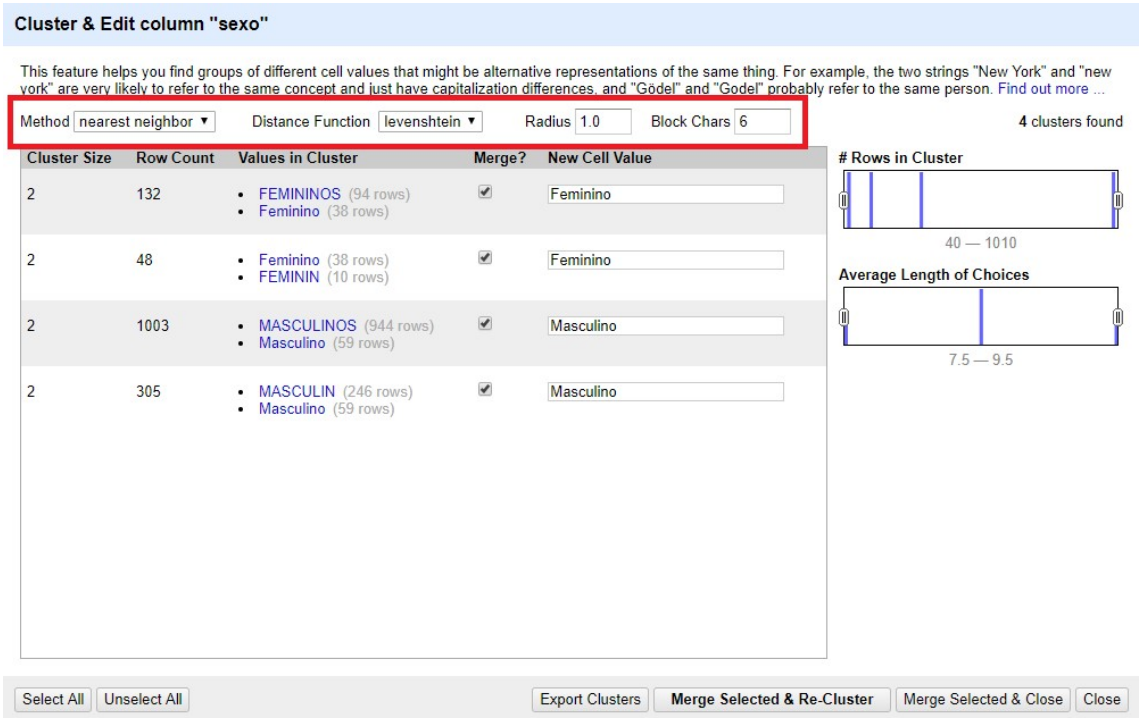


Figura 38 – Menu de *Clustering* do OpenRefine

Após a aplicação do algoritmo é necessário escolher quais os valores a uniformizar.
Para alterar os valores de um campo em massa temos de clicar na seta do campo a modificar e escolher "Edit cells" e "Transform…" como se pode ver na figura 39.

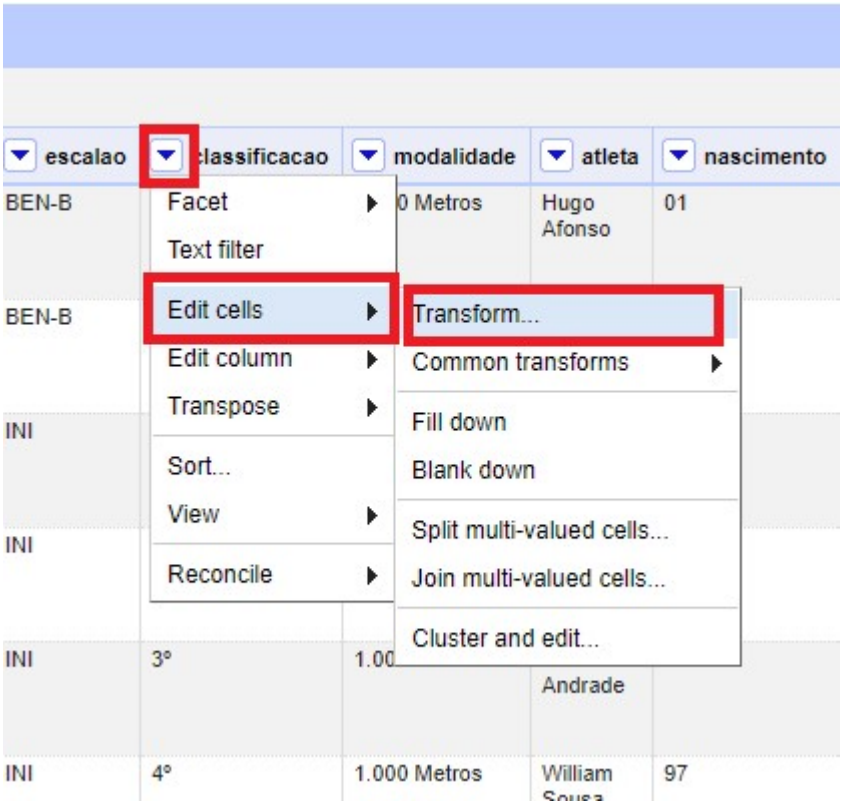


Figura 39 – Caminho para Aceder ao Menu de Transformação em Massa

Uma vez no menu fazem-se as alterações desejadas com a linguagem de programação Python ou GREL (General Refine Expression Language) e uma amostra do resultado é apresentada, tal como exemplificado na figura 40.

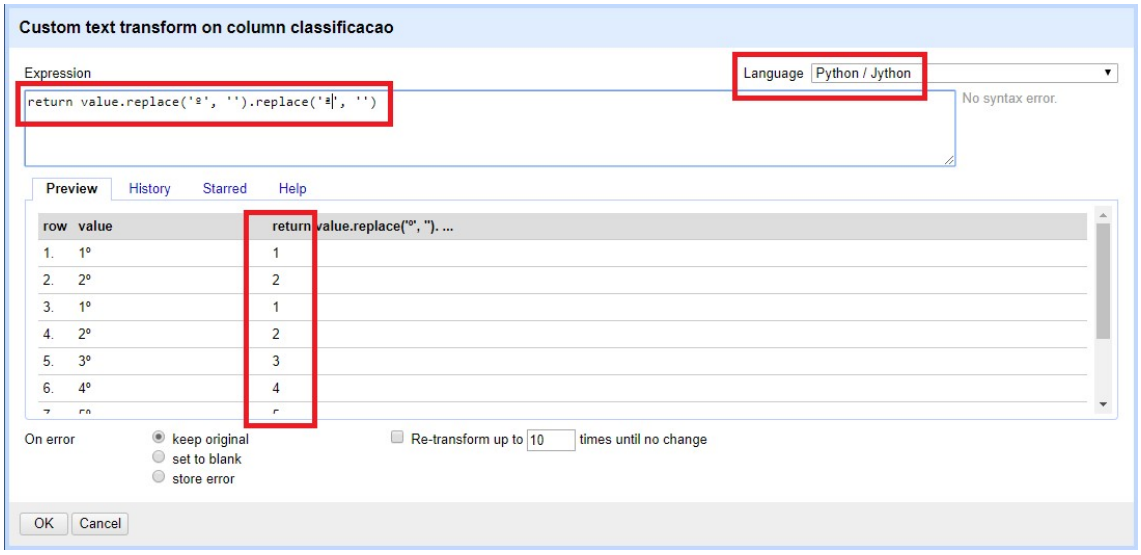


Figura 40 – Menu de Tranformação Programática dos Valores de um Campo

Também é possível criar valores de um campo com base em valores de outros campos recorrendo à linguagem GREL. Um exemplo é demonstrado na figura 41, onde valores para o campo das classificações são obtidos a partir do campo “classificacao_atleta”.

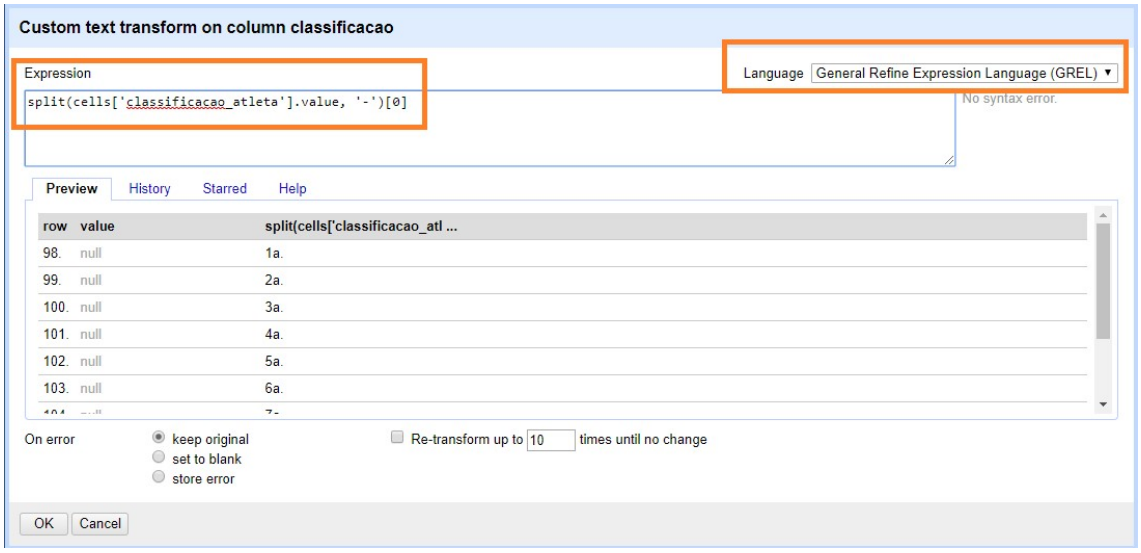


Figura 41 – Aceder Programaticamente a Valores de Outros Campos com GREL

4.9 Quinta Fase – Incremento dos Dados

Os dados das várias fontes estruturadas são recolhidos numa única fonte de ficheiros CSV tal como exemplificado na figura 42.

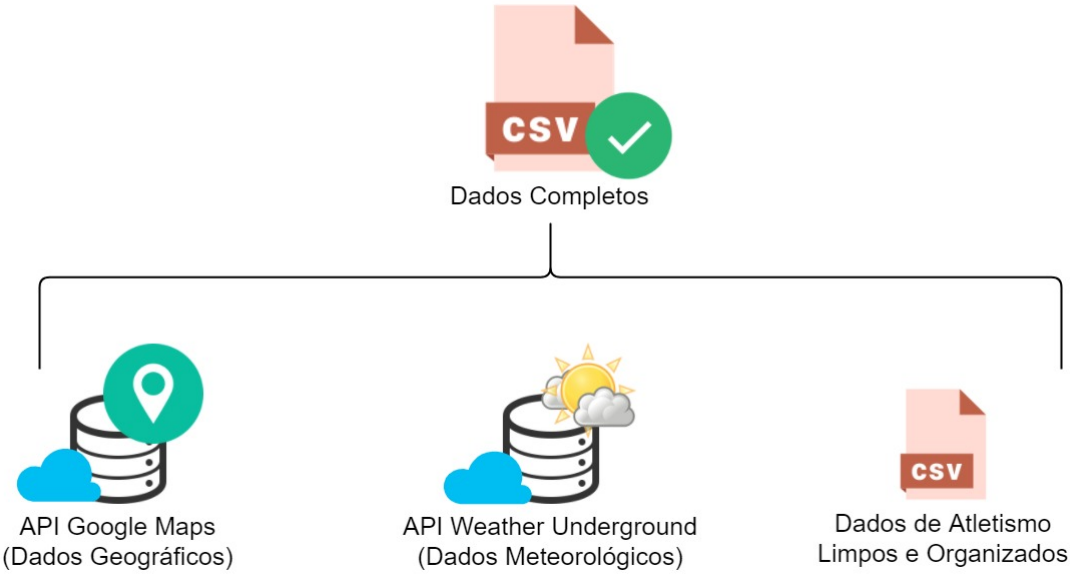


Figura 42 – Representação do Incremento dos Dados

Recorrendo ao módulo "pandas" e à ferramenta "Jupyter" foi construído um *script* Python capaz de obter novos dados a partir dos já existentes. Um exemplo do código para obter dados geográfico a partir do local das provas é apresentado abaixo:

```
# obter coordenadas
locais = df['local'].unique()
for local in locais:
    # obter latitude e longitude
```

```
    url_gps =
      ('https://maps.googleapis.com/maps/api/geocode/json?address='
       + local
       + '&key=AIzaSyBk967-NoF84g-R5CbGEpRjblUyrzrMQGA')
    resposta = requests.get(url_gps)
    resposta_json = resposta.json()
    latitude =
      resposta_json['results'][0]['geometry']['location']['lat']
    df.loc[df['local'] == local, 'latitude'] = latitude
    longitude =
      resposta_json['results'][0]['geometry']['location']['lng']
    df.loc[df['local'] == local, 'longitude'] = longitude
    # obter altitude
    url_altitude =
     ('https://maps.googleapis.com/maps/api/elevation/json?locations='
      + str(latitude) + ',' + str(longitude)
      + '&key=AIzaSyAHvptEZVGbUMK9rxPV58Asc8tDIBns6FY')
    resposta = requests.get(url_altitude)
    resposta_json = resposta.json()
    altitude = resposta_json['results'][0]['elevation']
    df.loc[df['local'] == local, 'altitude'] = altitude
print(df.head())
```

O primeiro URL é para obter a latitude e longitude onde "address" é o nome do local e "key" é a chave da conta utilizada. O segundo URL é referente à altitude do local onde a variável "locations" (a ser fornecida) é referente à sua latitude e longitude. Para obter os dados meteorológicos o processo é semelhante.
Uma descrição dos URLs das APIs e retorno da informação pode ser visto nas tabelas 9, 10, 11 e 12.

API	Google Maps Geocoding	
Limites	Limite de 2500 pedidos por dia	
URL	https://maps.googleapis.com/maps/api/geocode/json?address=ENDEREÇO &key=CHAVE_API	
Exemplo URL	https://maps.googleapis.com/maps/api/geocode/json?address=Mirandela& key= AIzaSyBk967-NoF84g-R5CbGEpRjblUyrzrMQGA	
O que faz?	Obtém coordenadas GPS a partir do nome do local	
O que retorna?	Latitude	Caminho: resultados['results'][0]['geometry']['location']['lat']
	Longitude	Caminho: resultados['results'][0]['geometry']['location']['lng']

Tabela 9 – Detalhes da Utilização da API Google Maps Geocoding

API	Google Maps Elevation	
Limites	Limite de 2500 pedidos por dia	
URL	https://maps.googleapis.com/maps/api/elevation/json?locations=LATITUDE ,LONGITUDE&key=CHAVE_API	
Exemplo URL	https://maps.googleapis.com/maps/api/elevation/json?locations=45.34324, 54.34324&key= AIzaSyBk967-NoF84g-R5CbGEpRjblUyrzrMQGA	
O que faz?	Obtém a altitude a partir das coordenadas GPS	
O que retorna?	Altitude	Caminho: resultados['results'][0]['elevation']

Tabela 10 – Detalhes da Utilização da API Google Maps Elevation

API	Weather Underground API (Geolookup)	
Limites	Limite de 500 pedidos por dia	
	Limite de 10 pedidos por minuto	
URL	http://api.wunderground.com/api/CHAVE/geolookup/q/LATITUDE,LONGITUDE.json	
Exemplo URL	http://api.wunderground.com/api/ f9d484cc23209a99/geolookup/q/45.34324, 54.34324.json	
O que faz?	Obtém o código geográfico a partir das coordenadas GPS	
O que retorna?	Código Geográfico	Caminho: resultados['location']['l']

Tabela 11 – Detalhes da Utilização da API Weather Underground (geolookup)

API	Weather Underground API (Dados Históricos)	
Limites	Limite de 500 pedidos por dia	
	Limite de 10 pedidos por minuto	
URL	http://api.wunderground.com/api/CHAVE_API/history_AAAAMMDD/q/PAIS/CIDADE.json	
Exemplo URL	http://api.wunderground.com/api/ f9d484cc23209a99/history_20170212/ q/zmw:00000.302.08570.json	
O que faz?	Obtém informação meteorológica histórica a partir do dia, mês, ano e código geográfico.	
O que retorna?	Temperatura (ºC)	Caminho: resultados['history']['dailysummary'][0]['meantempm']
	Humidade (%)	Caminho: resultados['history']['dailysummary'][0]['humidity']
	Pressão Atmosférica (hPa)	Caminho: resultados['history']['dailysummary'][0]['meanpressurem']
	Visibilidade (Km)	Caminho: resultados['history']['dailysummary'][0]['meanvism']
	Velocidade do Vento (Km/h)	Caminho: resultados['history']['dailysummary'][0]['meanwindspdm']
	Direção do Vento (North, South, etc)	Caminho: resultados['history']['dailysummary'][0]['meanwdire']
	Precipitação (mm)	Caminho: resultados['history']['dailysummary'][0]['precipm']
	Nevoeiro (0/1)	Caminho: resultados['history']['dailysummary'][0]['fog']
	Chuva (0/1)	Caminho: resultados['history']['dailysummary'][0]['rain']
	Neve (0/1)	Caminho: resultados['history']['dailysummary'][0]['snow']
	Pedraço (0/1)	Caminho: resultados['history']['dailysummary'][0]['hail']
	Trevoada (0/1)	Caminho: resultados['history']['dailysummary'][0]['thunder']
	Tornado (0/1)	Caminho: resultados['history']['dailysummary'][0]['tornado']
	Ponto de Condesação (ºC)	Caminho: resultados['history']['dailysummary'][0]['meandewptm']

Tabela 12 – Detalhes da Utilização da API Weather Underground (Dados Históricos)

4.10 Sexta Fase – Carregamento dos Dados

O modelo a aplicar no DW é o esquema em estrela onde a tabela central de factos é composta pelas marcas dos atletas, rodeada por 5 tabelas de dimensões: evento, meteorologia, modalidade, atleta e a dimensão temporal.

A estrutura em estrela do DW está representada na figura 43.

A dimensão temporal será a de maior relevo e tem o dia como o nível de granularidade mais baixo.

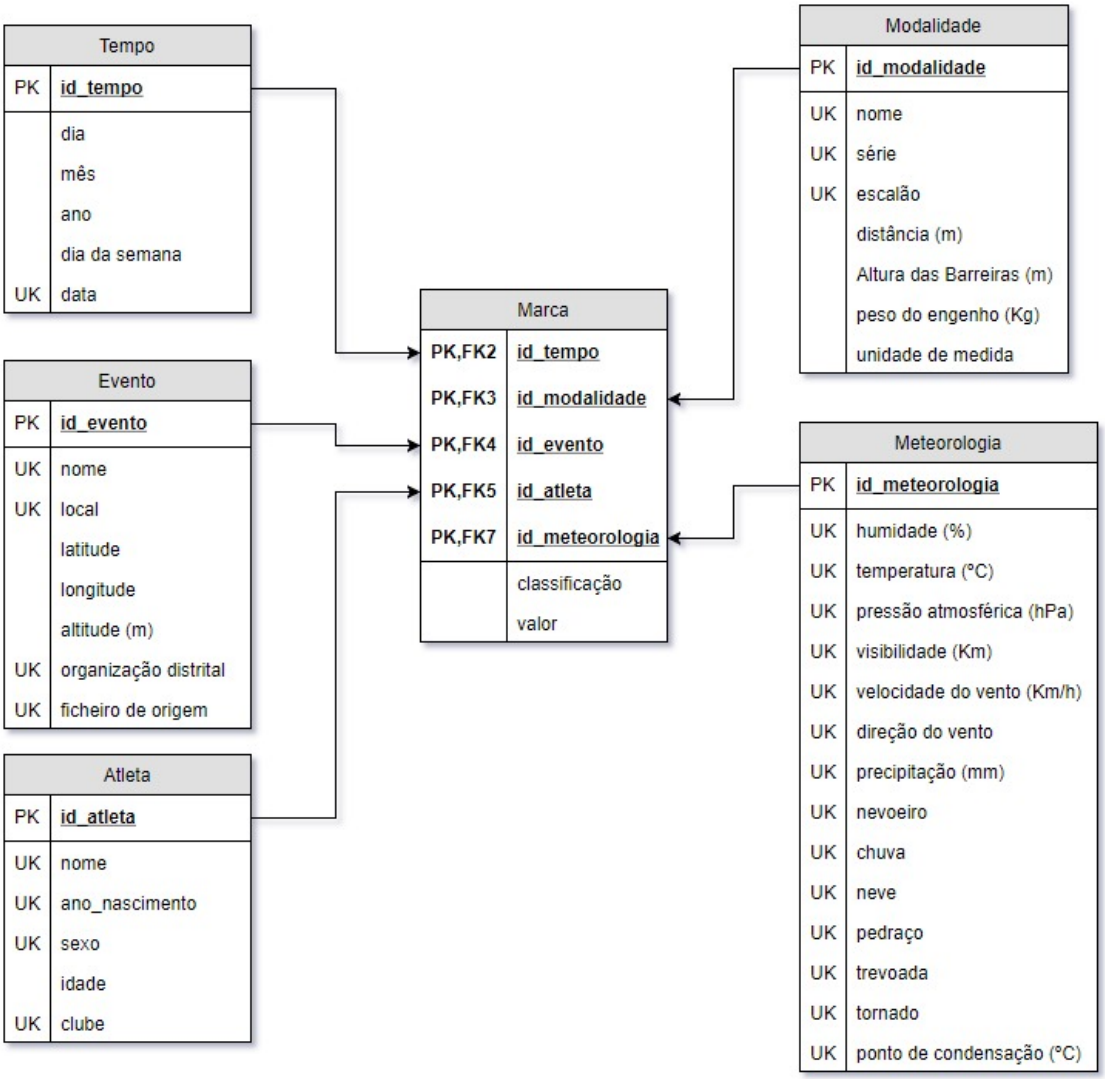


Figura 43 – Esquema em Estrela do DW

Com a ajuda da ferramenta de gestão de bases de dados "phpMyAdmin" e o servidor Web "Apache" é criado o DW em MySQL tal como observado na figura 44.

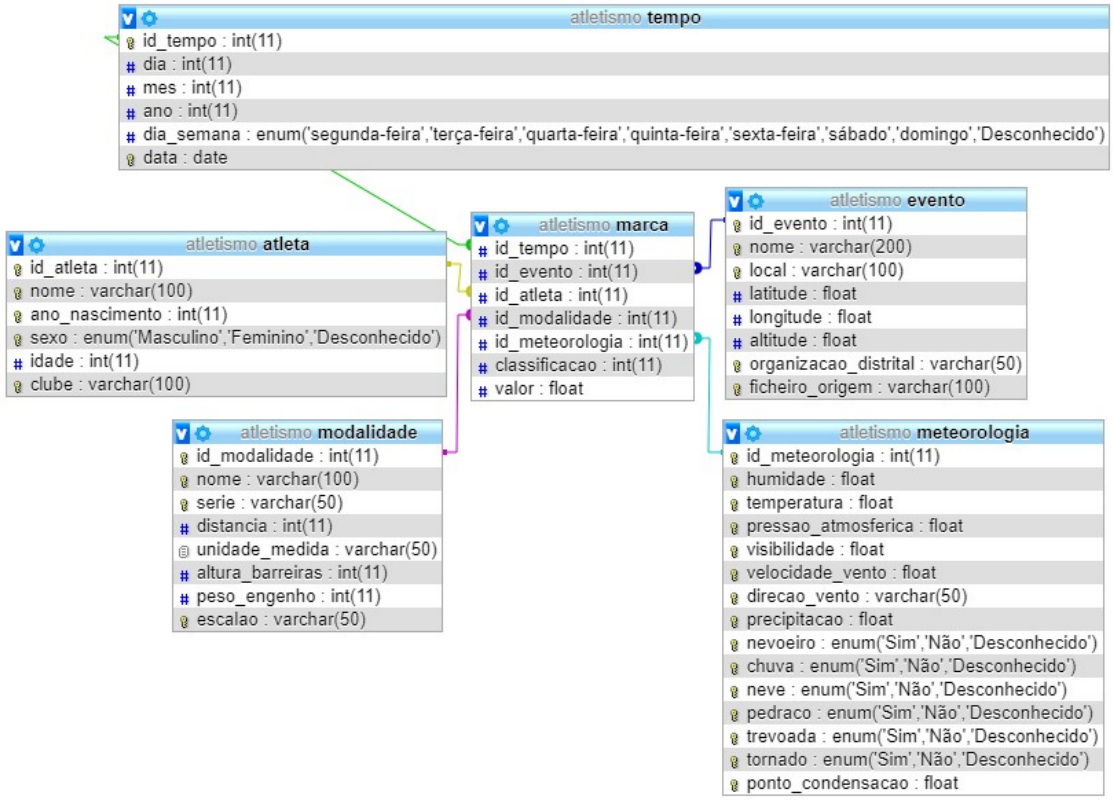


Figura 44 – Visualização do DW no phpMyAdmin

Para a quantidade de dados analisados neste projeto, a utilização de um esquema em estrela não é absolutamente necessária, no entanto este modelo foi adotado para o caso de mais dados serem adicionados ao longo do tempo e a partir de outras fontes de informação.

Depois da criação das tabelas o procedimento seguinte consiste em carregar os dados das dimensões com ajuda do módulo "pandas" em Python. Para carregar uma dimensão é preciso criar uma *data frame* com os seus campos e carregá-los para uma tabela temporária através do método "to_sql", depois usar um pedido SQL para transferir os dados da tabela temporária para a tabela permanente:

```
df_tempo = df_principal[['data', 'dia', 'mes', 'ano', 'dia_semana']]
motor_mysql =
create_engine('mysql+mysqldb://root:@localhost/atletismo')
df_tempo.drop_duplicates().to_sql(con=motor_mysql, name='_temp',
if_exists='replace', index=False)
conexao = motor_mysql.connect()
conexao.execute('INSERT INTO tempo(dia, mes, ano, dia_semana, data) '
                'SELECT dia, mes, ano, dia_semana, data FROM _temp '
                'ON DUPLICATE KEY UPDATE tempo.data=_temp.data')
conexao.execute('DROP TABLE _temp')
motor_mysql.dispose()
```

O "pandas" não possui uma forma de evitar que se insiram duplicados na BD, dando erro e não carregando nenhuma entrada se for definida uma chave única no MySQL ou adicionando duplicados, caso existam, se não for definida uma chave única. Daí a

utilização duma tabela temporária inicial para o carregamento dos dados (https://stackoverflow.com/questions/24879156/pandas-to-sql-with-sqlalchemy-duplicate-entries-error-in-mysqldb/29614207#29614207. Acedido em 4 de Fevereiro de 2018).

No pedido SQL é usada a expressão "ON DUPLICATE KEY UPDATE" e não a expressão mais óbvia "REPLACE" porque esta última faz internamente um "DELETE" seguido de um "INSERT", o que neste caso pode causar problemas, pois como o DW do projeto está em formato estrela, a tabela de factos possui uma grande quantidade de chaves estrangeiras e, em caso de duplicados, o processo iria criar uma nova entrada aparentemente igual mas com o ID da chave principal diferente, anulando possíveis ligações que a tabela de factos possa ter com as tabelas de dimensões. Pior ainda, se a relação entre as duas tabelas for em cascata (CASCADE) poderá indiretamente apagar dados da tabela de factos (https://stackoverflow.com/a/9168948/1778483. Acedido em 4 de Fevereiro de 2018). No entanto, utilizando a expressão "ON DUPLICATE KEY UPDATE" o campo de uma tabela é atualizado pelo campo análogo da tabela temporária com o mesmo valor, não devolvendo qualquer erro.

Hoje em dia, a grande maioria das bases de dados permite que múltiplas linhas tenham o valor "NULL" numa coluna "UNIQUE", seguindo a lógica de que dois nulos não são necessariamente iguais, pelo contrário, é mais provável que não o sejam (https://stackoverflow.com/questions/1346765/unique-constraint-that-allows-empty-values-in-mysql/1346776#1346776. Acedido em 4 de Fevereiro de 2018) (https://dev.mysql.com/doc/refman/5.5/en/create-table.html. Acedido em 4 de Fevereiro de 2018). Logo a utilização da expressão "ON DUPLICATE KEY UPDATE" não irá impedir a adição de duplicados com NULL nas colunas de chave única. No entanto, no caso concreto aqui explicado, é mais provável que essas linhas repetidas sejam a mesma entidade e por esse motivo os valores nulos (NULL) para referir campos desconhecidos ou em falta deverão ser substituídos por outro valor representativo de um campo em branco, antes de serem carregados. Neste caso essa substituição terá a seguinte correspondência:

- NULL de inteiros e floats → -9999;
- NULL de strings e enums → 'Desconhecido';
- NULL de datas → 0000-00-00.

Como pode ser observado no código utilizado:

```
# substituir campos de valor inteiro e float
campos_num = ['dia', 'mes', 'ano', 'classificacao', 'marca',
              'nascimento', 'idade', 'distancia', 'altura_barreiras',
              'peso_engenho', 'humidade', 'temperatura',
              'pressao_atmosferica', 'visibilidade',
              'velocidade_vento', 'precipitacao', 'ponto_condensacao',
              'latitude', 'longitude', 'altitude']
df_principal[campos_num]=df_principal[campos_num].fillna(-9999)

# substituir campos de valor string e enum
campos_string = ['direcao_vento', 'prova', 'local',
```

```
                  'organizacao_distrital', 'File', 'atleta', 'clube',
                  'modalidade', 'serie', 'unidade_medida', 'escalao',
                  'dia_semana', 'nevoeiro', 'chuva', 'neve', 'pedraco',
                  'trevoada', 'tornado', 'sexo']
df_principal[campos_string] = df_principal[campos_string].fillna(
                                    'Desconhecido')

# substituir campos de valor data
campos_data = ['data']
df_principal[campos_data]=df_principal[campos_data].fillna(
    '0000-00-00')
```

O procedimento correto para carregar novos dados num esquema em estrela é primeiro introduzir os dados das dimensões e só depois os dados correspondentes da tabela de factos (https://it.toolbox.com/question/etl-load-sequence-for-a-star-schema-042208. Acedido em 4 de Fevereiro de 2018).

4.11 Sétima Fase – BI

Para correr a ferramenta de BI "Metabase" basta, pela linha de comandos na diretoria do seu ficheiro java, introduzir o seguinte comando:

```
java –jar metabase.jar
```

Tal como demonstrado na figura 45.

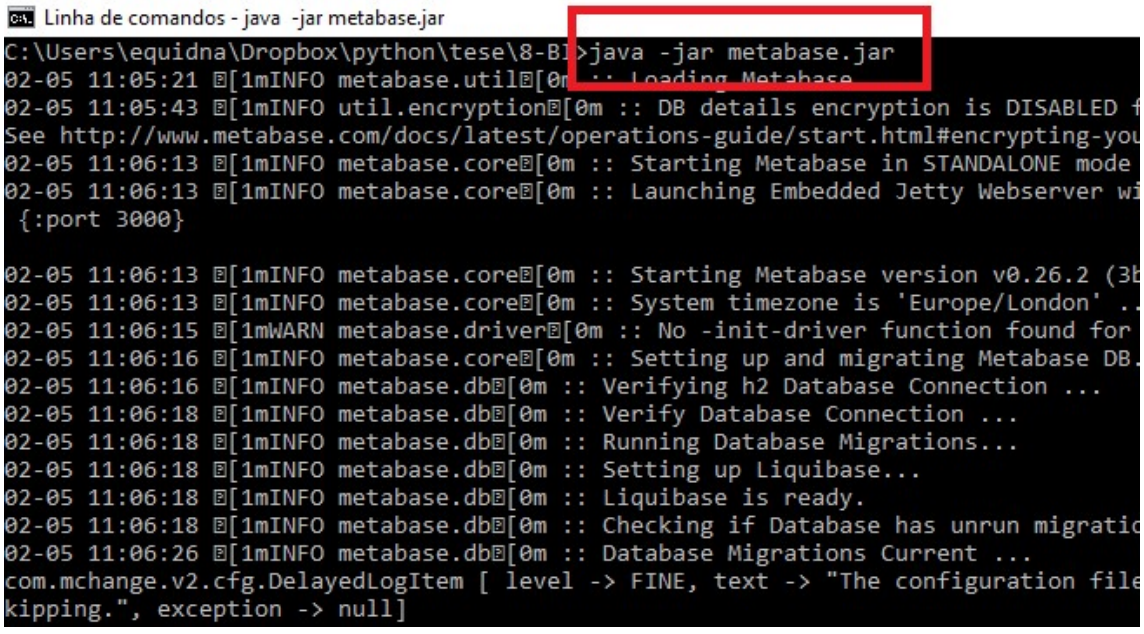


Figura 45 – Lançamento da Aplicação Metabase

Quando a aplicação é lançada pela primeira vez é necessário fazer a sua configuração onde são definidos o *email* e palavra passe do administrador que mais tarde são usados para o seu acesso, como demonstrado na figura 46.

Figura 46 – Página de Login do DW

É acedido o painel de administração para carregar a base de dados e preparar os meta dados das tabelas, para isso é necessário abrir o menu do canto superior direito e clicar em "Admin Panel" por um utilizador com permissões de administrador, como é possível observar na figura 47.

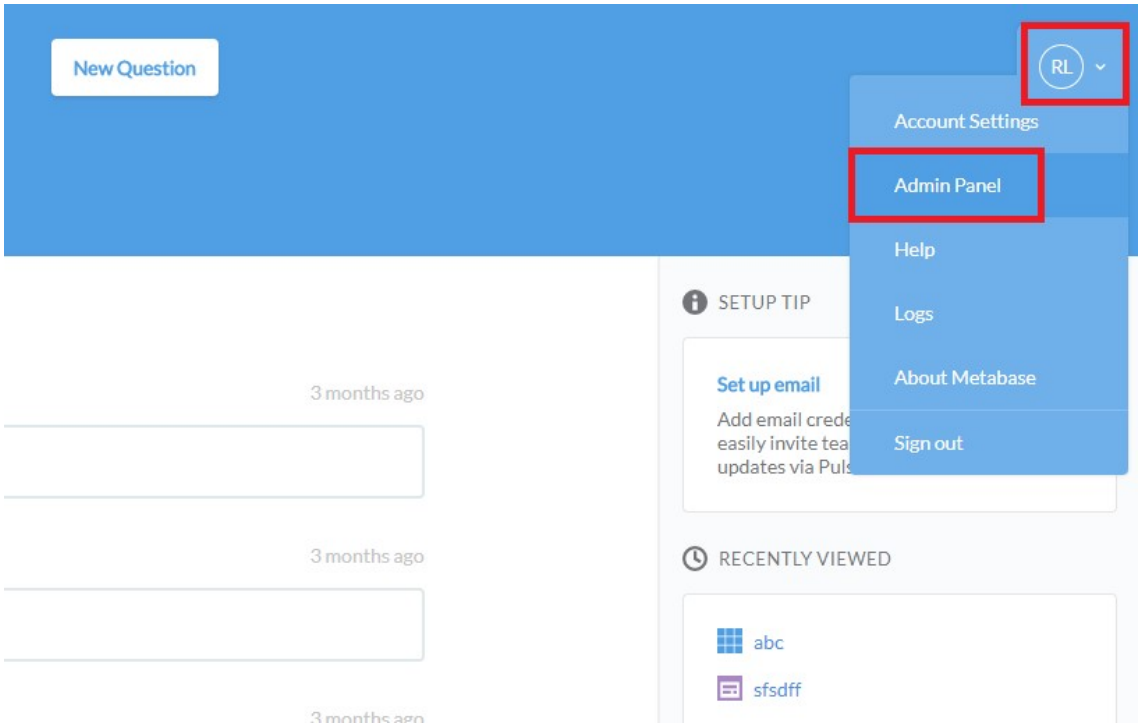


Figura 47 – Acesso ao Painel de Administração

Para adicionar uma base de dados basta clicar em "Add database" na opção de menu "Databases", visto na figura 48, e adicionar a sua informação.

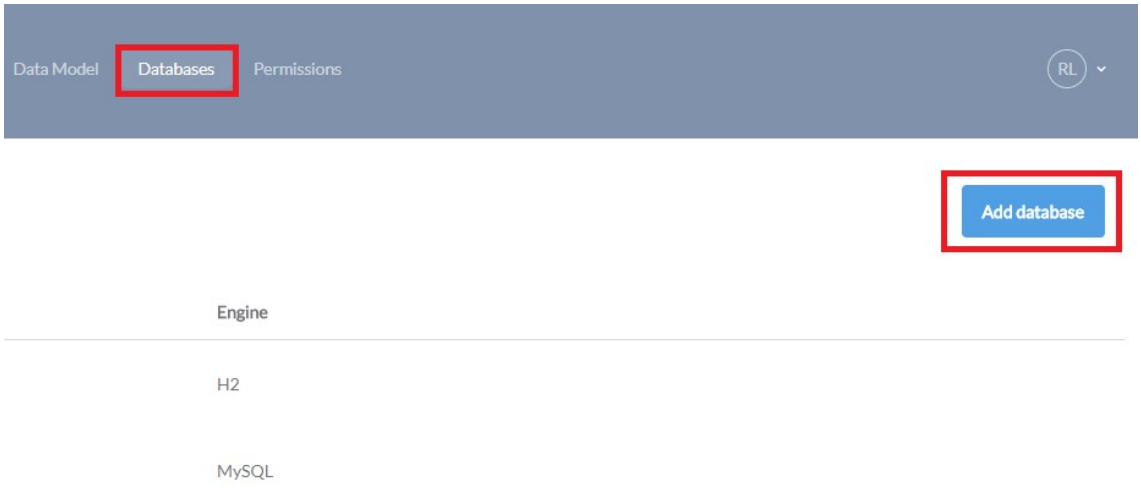


Figura 48 – Acrescentar uma Base de Dados ao DW

Os nomes das tabelas e campos são alterados para linguagem natural na opção de menu "Data Model". A ferramenta mostra uma barra para cada tabela que mede a quantidade de meta dados já completos (ver figura 49).

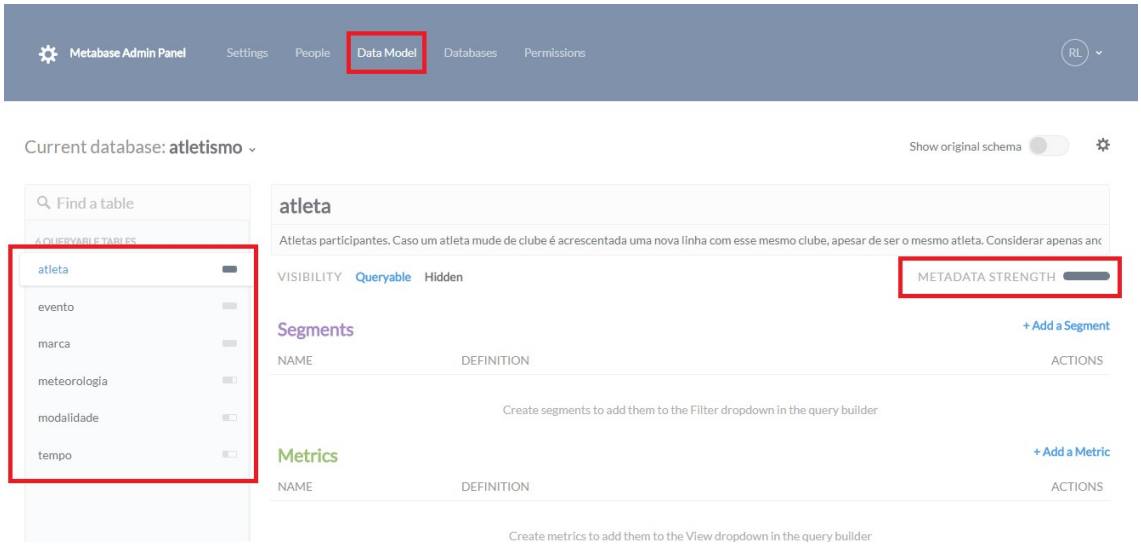


Figura 49 – Adicionar e Editar Meta Dados

Qualquer utilizador mesmo sem permissões de administrador, pode gerar gráficos com consultas SQL. A forma mais fácil de o fazer é clicando na opção "New Question" e em seguida no botão "Custom" (figura 50) onde uma pergunta pode ser feita para gerar a informação desejada e um gráfico pode ser escolhido para a representar, como demonstrado na figura 51.

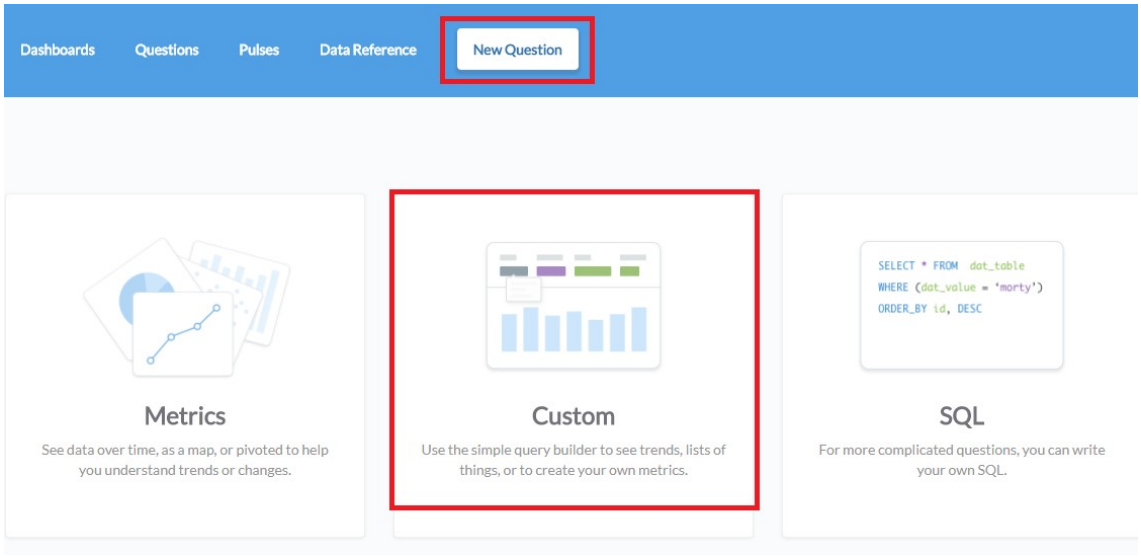


Figura 50 – Acesso ao Menu de Perguntas

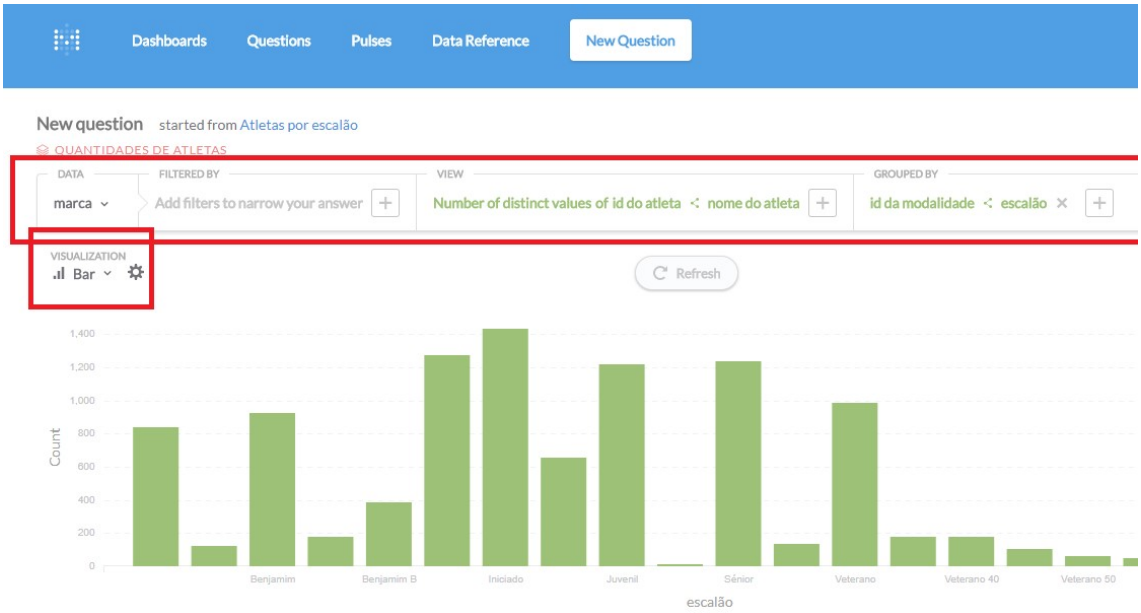


Figura 51 – Gerar uma Nova Pergunta

As questões podem ser agrupadas em coleções. Para criar uma nova coleção é necessário clicar em “New Collection” na opção “Collections”, como pode ser observado na figura 52.

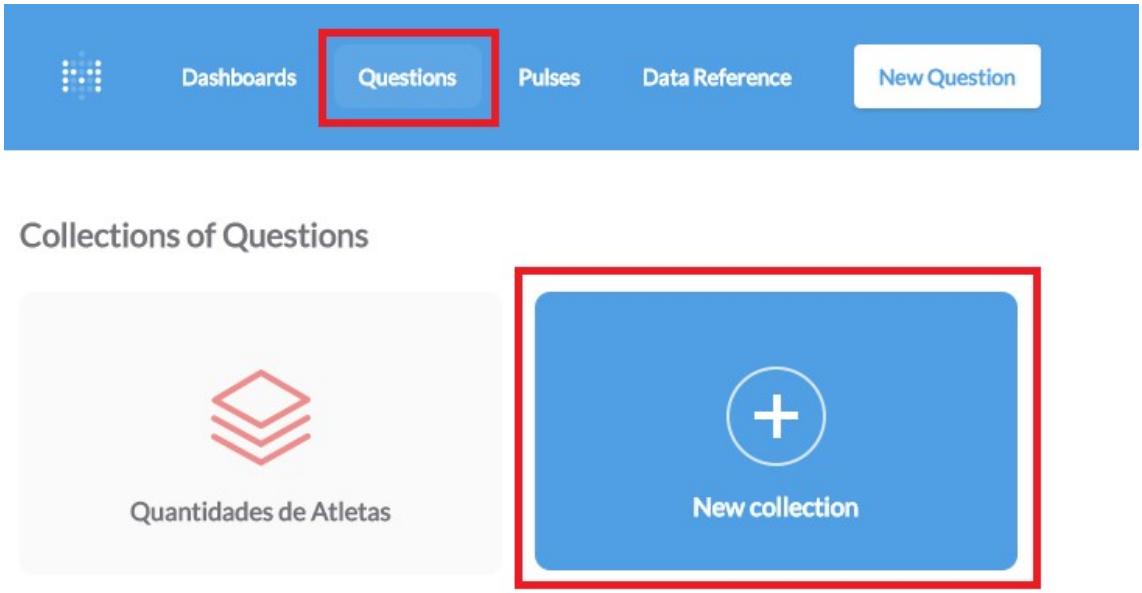


Figura 52 – Criar uma Nova Coleção de Dados

Os gráficos gerados podem ser agrupados em painéis para um rápido acesso, bastando para isso clicar em “Dashboards” e definir o seu nome, como se pode ver na figura 53.

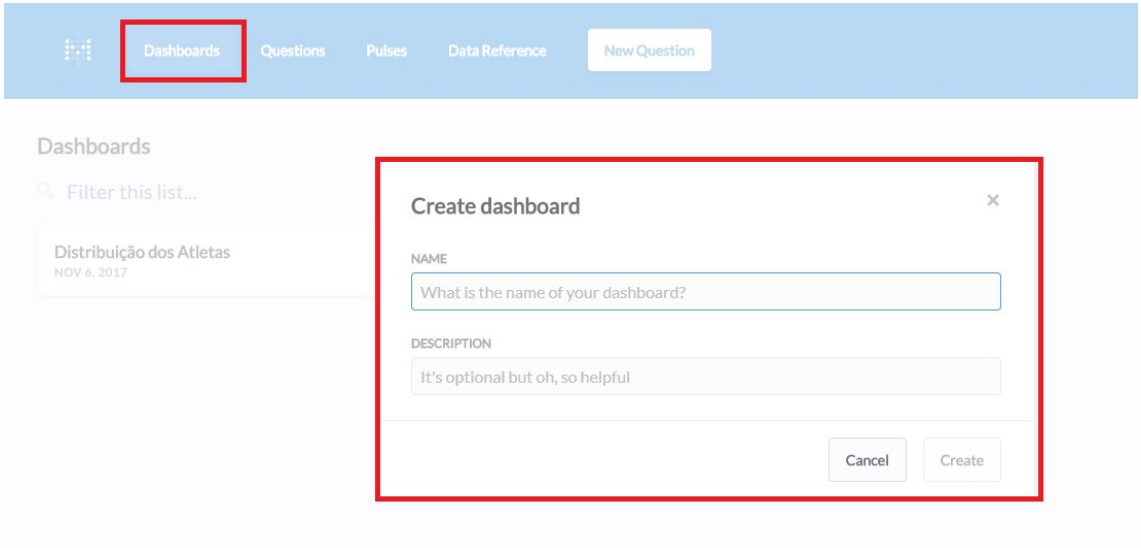


Figura 53 – Criação de um Novo Painel

Na figura 54 é possível observar os gráficos apresentados num dos painéis criados.

Figura 54 – Painel com Gráficos Gerados a Partir do DW

4.12 OITAVA FASE – DM

Foi utilizado o "Orange Canvas" como ferramenta de DM. Esta ferramenta permite analisar os dados de forma prática e intuitiva recorrendo apenas aos módulos que o utilizador achar necessários para cada pesquisa, chamados de "widgets", que podem ser ligados entre si. Cada ligação representa a saída dos dados de um "widget" e entrada de dados para outro e o fluxo de dados é atualizado em tempo real sempre que algo seja alterado.

O fluxo de dados criado neste projeto de investigação, pode ser visto na figura 55, e envolve quatro fases: a preparação dos dados, uma análise exploratória, testes de algoritmos e previsões.

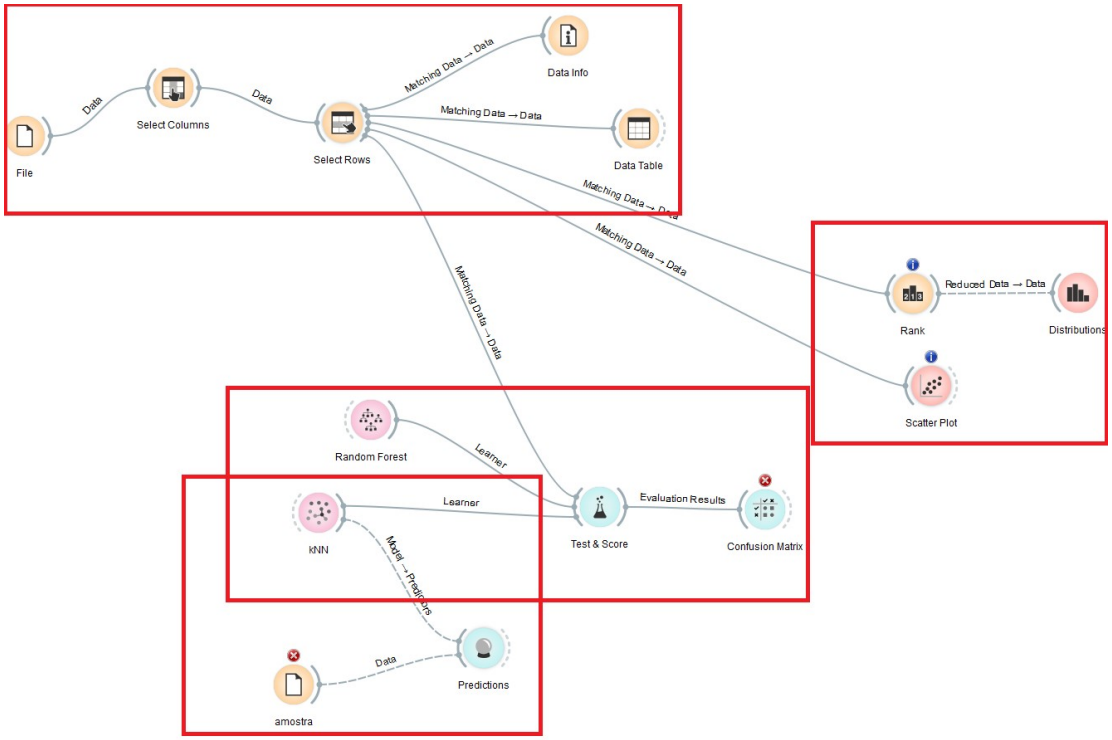


Figura 55 – Painel Principal da Ferramenta Orange Canvas

Na primeira fase foi criado um "widget" de importação de ficheiros, para obter os dados (File), e definida como caraterística alvo (campo a obter informações) a marca dos atletas. Filtraram-se também as colunas e linhas desejadas (Select Columns e Select Rows) e acrescentou-se um visualizador para dados resultantes (Data Table) e um resumo da informação dos dados (Data Info). Esta fase pode ser vista ao pormenor na figura 56.

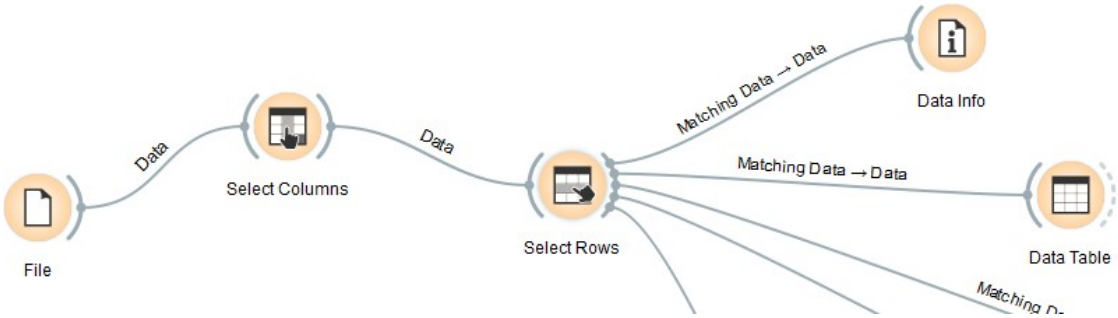


Figura 56 – Fluxo de Preparação dos Dados

Antes de testar os métodos de DM propriamente dito, efetuou-se uma breve exploração dos dados recorrendo aos *widgets* "Rank", "Distributions" e "Scatter Plot", como se pode ver na figura 57.

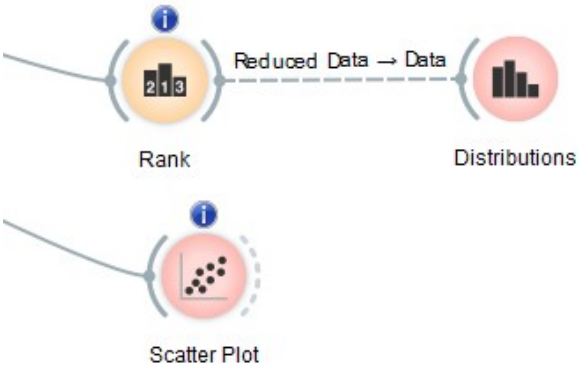


Figura 57 – Fluxo da Análise Exploratória dos Dados

Foram testados 2 algoritmos de DM ("Random Forest" e "kNN") utilizando o módulo de testes ("Test & Score") e uma matriz de confusão ("Confusion Matrix"), demonstrado na figura 58.

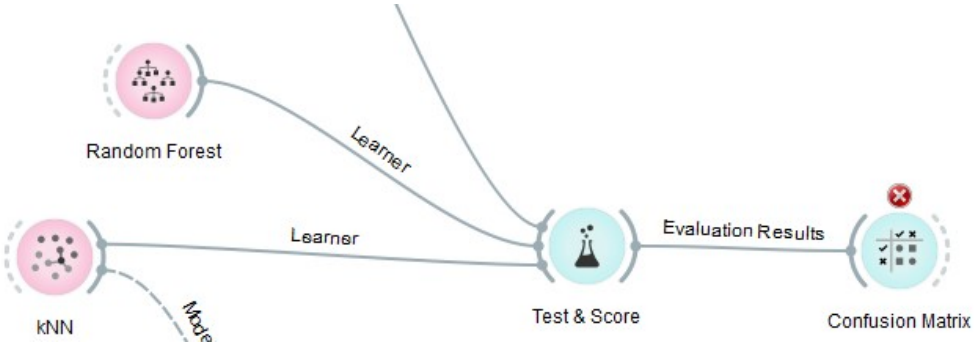


Figura 58 – Fluxo dos Testes de Algoritmos

Na última fase efetuam-se previsões com o algoritmo que obteve maior performance na fase de testes, face aos dados existentes, demonstrado na figura 59.

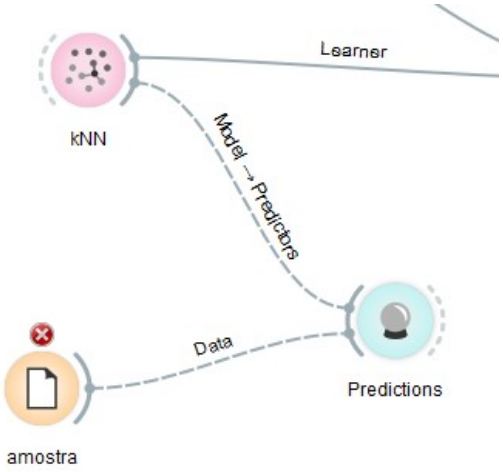


Figura 59 – Fluxo para a Previsão de Dados

5. ANÁLISE DOS RESULTADOS

Na análise dos resultados é feito primeiro um balanço geral dos ficheiros obtidos ao longo de todo o processo de obtenção e tratamento dos dados assim como um pequeno estudo efetuado com as ferramentas de BI e DM.

5.1 FASES DE 1 A 6 – TRATAMENTO DOS DADOS

Ao longo das fases de tratamento de dados existem distritos que fornecem uma grande quantidade de ficheiros ao passo que outros fornecem uma quantidade mais escassa. Também existem grandes diferenças quanto ao tipo de ficheiros, onde o PDF é, de longe, o tipo de ficheiro predominante.

Na tabela 13 é possível observar o número de ficheiros ao longo das três primeiras fases do trabalho (Web scraping, verificação de conteúdo e extração de conteúdo) descriminados por distrito. Faro é o distrito português com maior número de ficheiros válidos disponíveis na sua plataforma enquanto Bragança é o distrito com o menor número de ficheiros. O número total de ficheiros aumentou da fase de "Web Scraping" para a "Verificação de Conteúdo" porque alguns se encontravam comprimidos num só ficheiro ".rar" ou ".zip".

	Web Scraping	Verificação de Conteúdo	Extração de Conteúdo
Açores	522	524	478
Aveiro	703	1307	478
Beja	543	548	276
Braga	1147	1147	572
Bragança	7	7	3
Castelo Branco	798	794	616
Coimbra	180	180	124
Évora	846	839	282
Faro	2306	2300	630
Guarda	925	925	348
Leiria	931	1112	396
Lisboa	303	299	194
Madeira	755	753	507
Portalegre	180	179	84
Porto	747	747	293
Santarém	508	525	159
Setúbal	767	186	378
Viana do Castelo	291	290	267
Vila Real	69	26	64
Viseu	27	26	24
Nacionais	336	336	195
total	12891	13050	6368

Tabela 13 – Evolução do Número de Ficheiros por Distrito

Esse número de ficheiros pode ser ainda descriminado pelo seu tipo, tal como exposto na tabela 14, 15 e 16. PDF é o formato mais utilizado na maior parte dos distritos.

	Web Scraping							
	pdf	html	jpeg/png	rar	zip	doc	txt	xls/xlsx
Açores	412	0	0	0	0	61	0	52
Aveiro	653	0	0	2	48	0	0	0
Beja	553	0	0	0	0	0	0	0
Braga	862	74	0	0	0	0	207	4
Bragança	1	0	0	0	0	0	6	0
Castelo Branco	443	0	0	0	0	0	354	1
Coimbra	180	0	0	0	0	0	0	0
Évora	846	0	0	0	0	0	0	0
Faro	2301	0	0	0	0	0	0	5
Guarda	921	0	0	0	4	0	0	0
Leiria	841	8	46	11	9	2	1	13
Lisboa	293	0	0	0	0	0	1	9
Madeira	732	4	1	0	0	0	0	18
Portalegre	178	2	0	0	0	0	0	0
Porto	668	7	0	0	0	1	0	71
Santarém	507	0	0	0	1	0	0	0
Setúbal	753	13	0	0	0	0	0	1
Viana do Castelo	291	0	0	0	0	0	0	0
Vila Real	67	1	0	1	0	0	0	0
Viseu	0	0	0	0	0	27	0	0
Nacionais	336	0	0	0	0	0	0	0
totais	11838	109	47	14	62	91	569	174

Tabela 14 – Número de Ficheiros por Distrito e por Tipo Obtidos com Web Scraping

	Verificação de Conteúdo				
	pdf	html	doc	txt	xls/xlsx
Açores	411	0	61	0	52
Aveiro	1306	0	0	0	1
Beja	548	0	0	0	0
Braga	862	0	74	207	4
Bragança	1	0	0	6	0
Castelo Branco	439	0	0	354	1
Coimbra	180	0	0	0	0
Évora	839	0	0	0	0
Faro	2295	0	0	0	5
Guarda	925	0	0	0	0
Leiria	1013	7	2	29	61
Lisboa	289	0	0	1	9
Madeira	731	4	0	0	18
Portalegre	177	2	0	0	0
Porto	668	7	1	0	71
Santarém	525	0	0	0	0
Setúbal	746	13	0	0	1
Viana do Castelo	291	0	0	0	0
Vila Real	73	0	0	0	0
Viseu	0	0	25	0	0
Nacionais	336	0	0	0	0
totais	12655	33	163	597	223

Tabela 15 – Número de Ficheiros por Distrito e por Tipo Após Verificação do Conteúdo

	Extração do Conteúdo			
	pdf	doc	txt	xls/xlsx
Açores	368	61	0	49
Aveiro	478	0	0	0
Beja	276	0	0	0
Braga	572	0	0	0
Bragança	1	0	2	0
Castelo Branco	284	0	332	0
Coimbra	124	0	0	0
Évora	282	0	0	0
Faro	630	0	0	0
Guarda	348	0	0	0
Leiria	395	0	0	1
Lisboa	194	0	0	0
Madeira	507	0	0	0
Portalegre	84	0	0	0
Porto	293	0	0	0
Santarém	195	0	0	0
Setúbal	159	0	0	0
Viana do Castelo	378	0	0	0
Vila Real	267	0	0	0
Viseu	64	0	0	0
Nacionais	0	24	0	0
totais	5899	85	334	50

Tabela 16 – Número de Ficheiros por Distrito e por Tipo Após Extração do Conteúdo

Através dos gráficos das figuras 60, 61 e 62 verifica-se uma discrepância entre os ficheiros PDF e os restantes formatos. Discrepância essa que se mantém ao longo das fases do projeto.

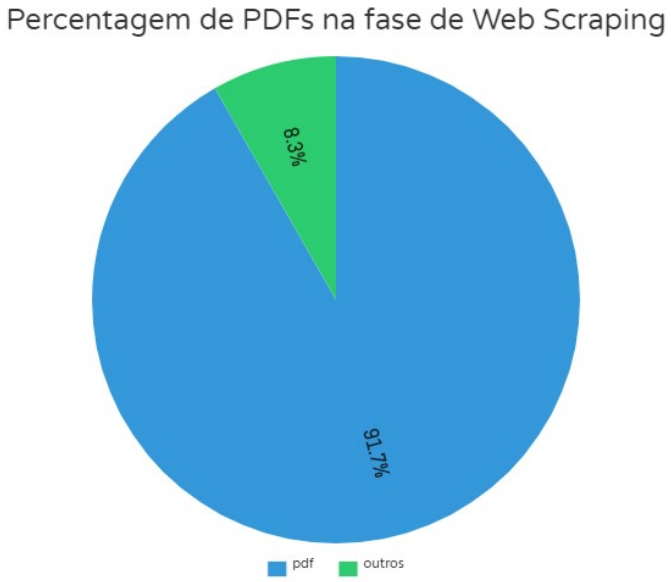


Figura 60 – Percentagem de PDFs Após Web Scraping

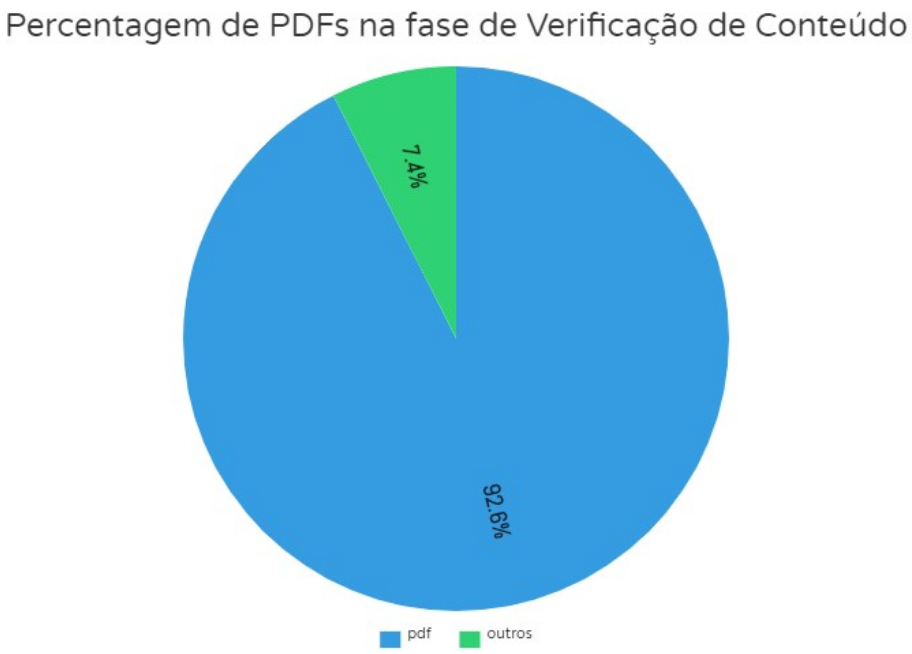


Figura 61 – Percentagem de PDFs Após a Verificação do Conteúdo

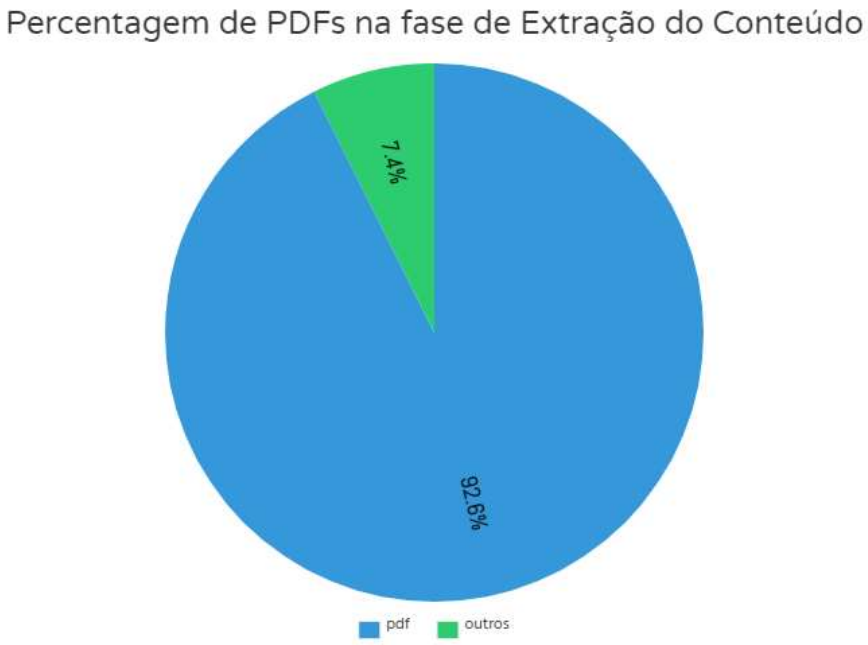


Figura 62 – Percentagem de PDFs Após a Extração do Conteúdo

Na tabela 17 verificam-se as linhas obtidas nas 4 últimas fases do tratamento de dados (extração de conteúdo, limpeza dos dados, incremento dos dados e carregamento dos dados). Na mesma tabela também se observam quais os distritos que foram completamente analisados e os distritos cuja análise não foi completamente feita por falta de tempo. Dos distritos analisados completamente, os açores foi o que acabou por fornecer um maior número de registos (69946).

	Extração de Conteúdo	Limpeza dos Dados	Incremento dos Dados	Carregamento dos Dados
Açores	86175	70017	70017	69946
Aveiro	66487	54618	54618	52355
Beja	35107	29777	29777	29592
Braga	200917	Inacabado	Inacabado	Inacabado
Bragança	1391	1391	1391	1375
Castelo Branco	46094	Inacabado	Inacabado	Inacabado
Coimbra	17547	Inacabado	Inacabado	Inacabado
Évora	51757	Inacabado	Inacabado	Inacabado
Faro	152907	Inacabado	Inacabado	Inacabado
Guarda	48709	Inacabado	Inacabado	Inacabado
Leiria	129753	Inacabado	Inacabado	Inacabado
Lisboa	54656	Inacabado	Inacabado	Inacabado
Madeira	176176	Inacabado	Inacabado	Inacabado
Portalegre	9102	Inacabado	Inacabado	Inacabado
Porto	91084	Inacabado	Inacabado	Inacabado
Santarém	47778	Inacabado	Inacabado	Inacabado
Setúbal	97347	Inacabado	Inacabado	Inacabado
Viana do Castelo	42845	Inacabado	Inacabado	Inacabado
Vila Real	11646	10598	10598	10517
Viseu	3108	2573	2573	2492
Nacionais	87915	Inacabado	Inacabado	Inacabado
total	1458501	168974	168974	166277

Tabela 17 – Número de Registos Por Distrito

Com a informação da tabela 17 é possível fazer uma estimativa da percentagem de informação existente no DW em relação à informação que existiria caso todos os distritos fossem analisados (demonstrado na figura 63).

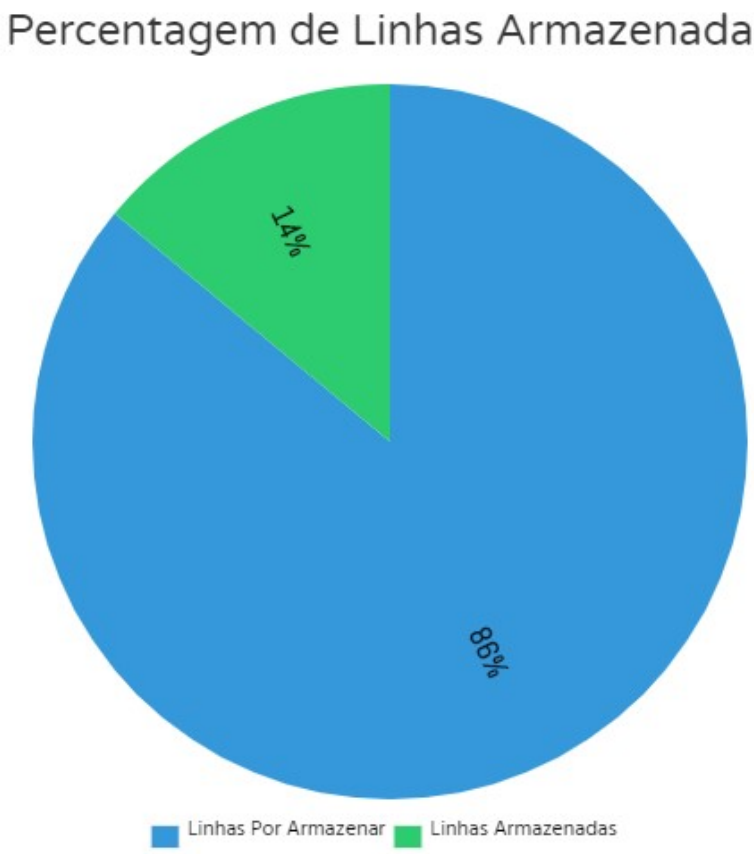


Figura 63 – Percentagem de Linhas Armazenadas e Por Armazenar Num DW

Na Tabela 18 é possível observar um resumo do total do número de registos, campos, ficheiros e distritos analisados ao longo das 6 fases de tratamento dos dados.

	Distritos Acabados	Linhas	Colunas	Ficheiros
Web Scraping	20	Não Aplicável	Não Aplicável	12904
Verificação de Conteúdo	20	Não Aplicável	Não Aplicável	13671
Extração do Conteúdo	20	1458501	13	6368
Limpeza dos Dados	6	168974	13	Não Aplicável
Incremento do Dados	6	168974	40	Não Aplicável
Carregamento dos Dados	6	166277	40	Não Aplicável

Tabela 18 – Breve Resumo das Fases de Tratamento de Dados

5.2 FASE 7 – BI

Recorrendo à ferramenta de BI "Metabase" é possível pesquisar a informação (dos distritos analisados) em diversos aspetos. No caso deste projeto, 3 áreas principais foram analisadas:

- A demografia dos atletas, como o seu género, escalão, modalidades praticadas e clubes;

- Fatores (precipitação e altitude) ambientais como possíveis influências do rendimento desportivo dos atletas;
- O percurso desportivo de um atleta em particular, como a evolução do rendimento de um determinado atleta numa determinada modalidade e mapear a localização das suas provas realizadas.

Na figura 64 observa-se a distribuição dos atletas segundo o seu género, onde se pode ver que cerca de 60% dos atletas são do género masculino e 40% do género feminino.

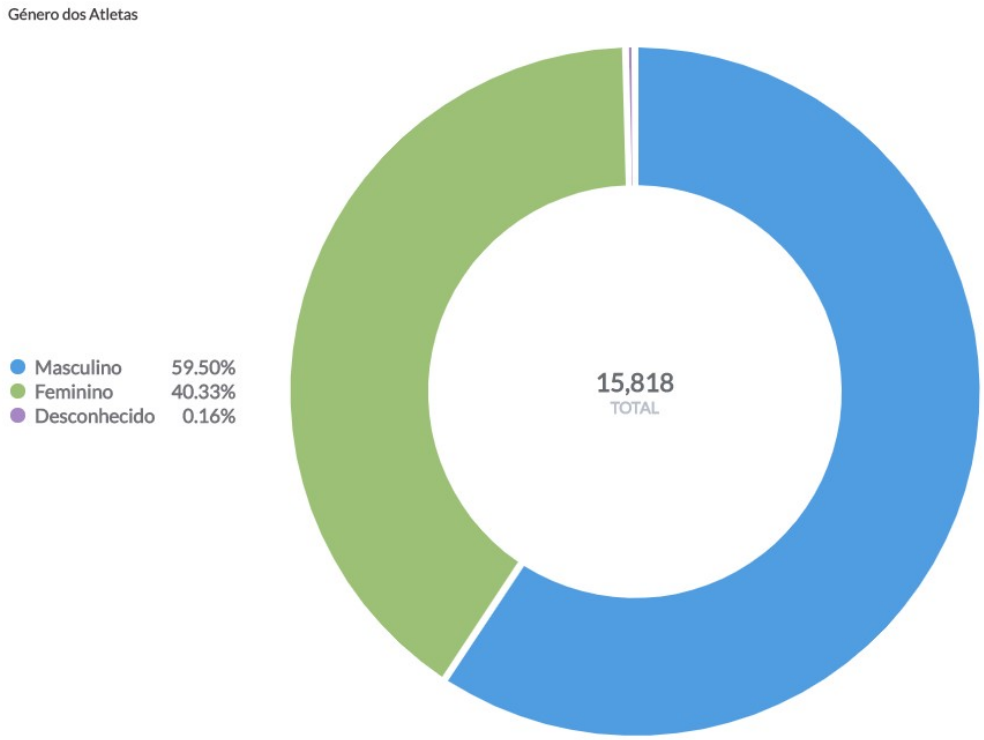


Figura 64 – Percentagem de Atletas Divididos Por Género

A distribuição dos atletas por escalão pode ser vista no gráfico apresentado na figura 64.

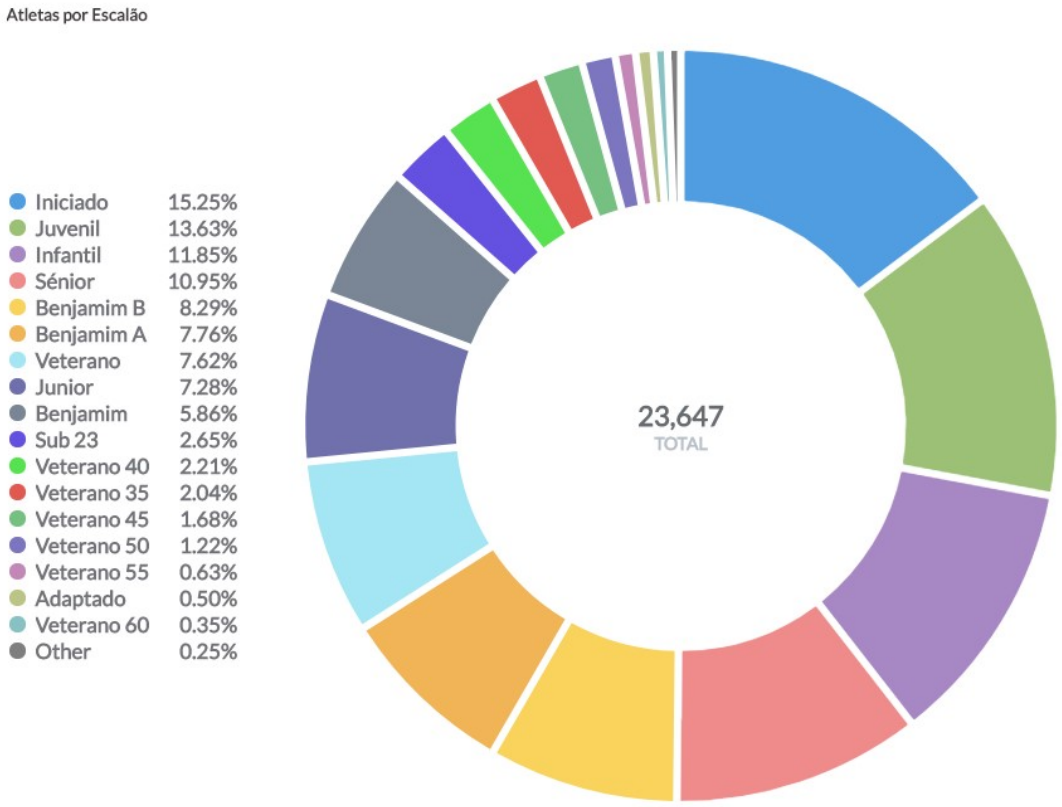


Figura 65 – Percentagem de Atletas Divididos Por Escalão

O escalão de veteranos tem início aos 35 anos e encontra-se subdividido em incrementos de 5 anos. Como se pode observar na figura 65, os escalões com maior percentagem de participantes são os de faixas etárias mais baixas (iniciados, infantil e juvenil) e uma faixa etária mais alta (sénior).

A figura 66 representa como as principais modalidades de atletismo estão distribuídas no que diz respeito à quantidade de participações de atletas. Da análise do gráfico apresentado na figura 66, pode-se concluir que as modalidades com maior número de participantes são a corrida, seguida de salto em comprimento e lançamento do peso.

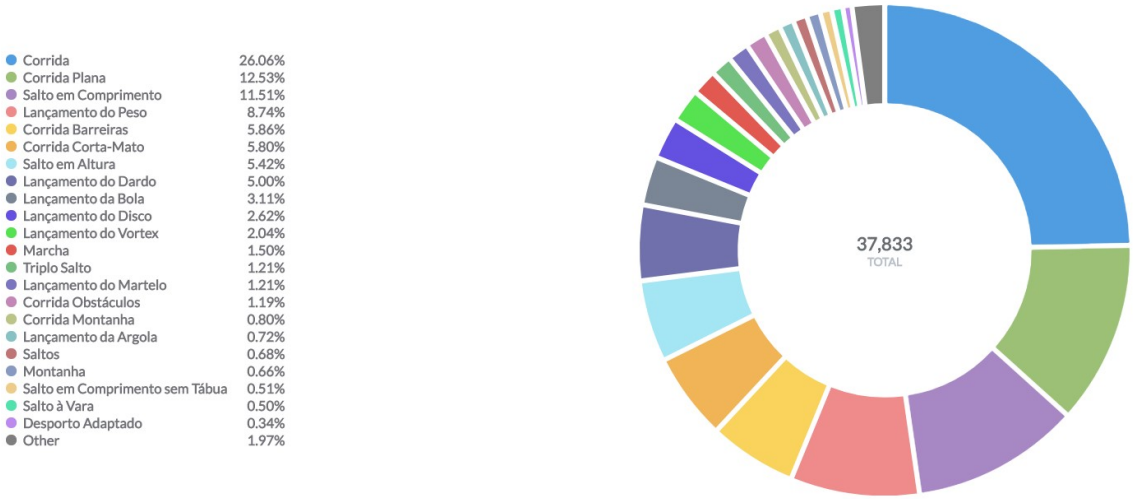


Figura 66 – Percentagem de Participações de Atletas Por Modalidade

Na figura 67 observa-se a distribuição dos 10 clubes com maior número de atletas participantes. Na figura 68 pode-se observar a evolução da quantidade de atletas distintos em cada ano.

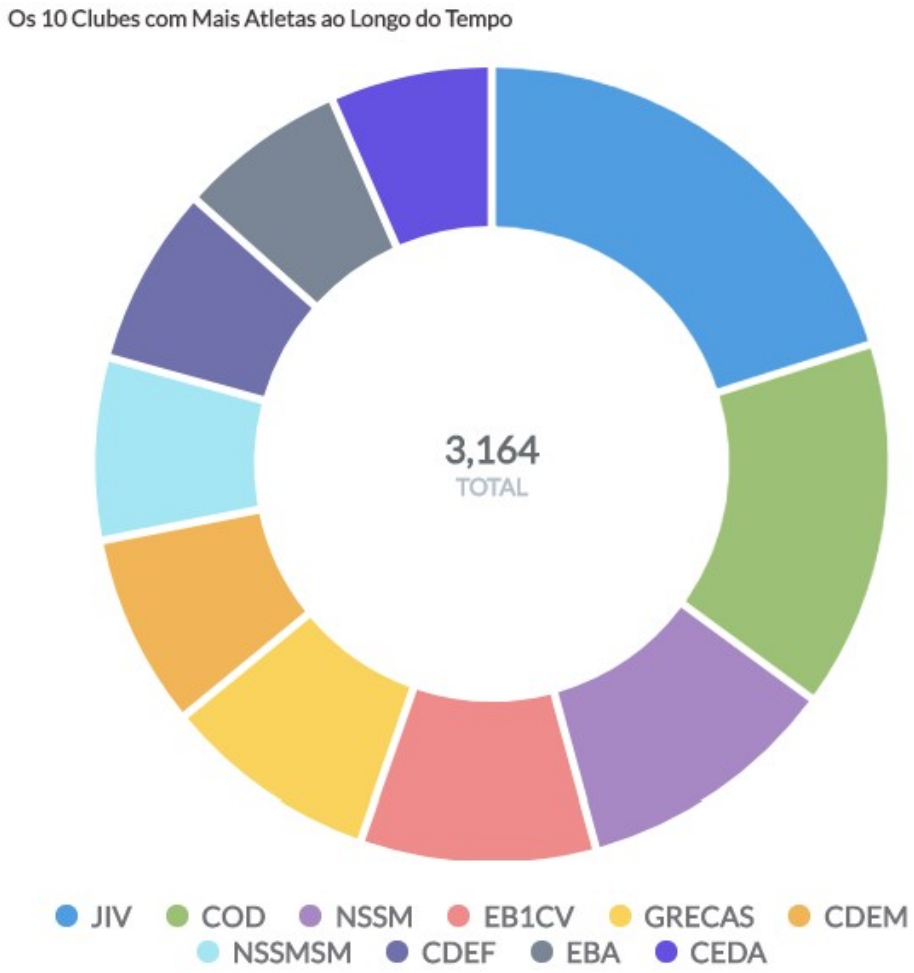


Figura 67 – Distribuição dos 10 Clubes Mais Populares

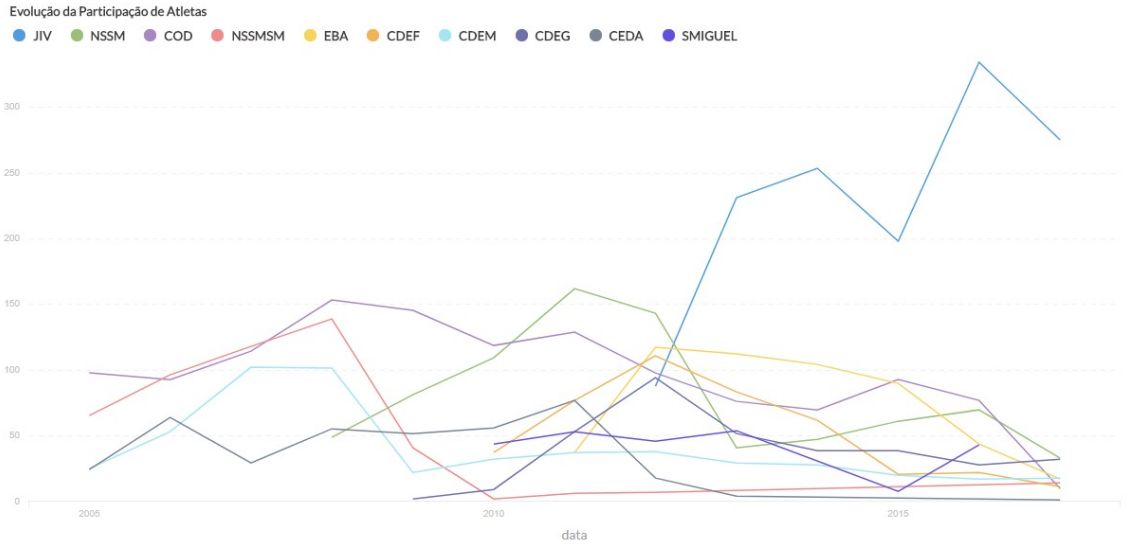


Figura 68 – Evolução da Popularidade dos 10 Clubes Mais Populares

Para além de observar os clubes historicamente mais populares, também é feita uma pesquisa quanto aos 10 clubes mais populares da atualidade (ano de 2017), tal como observado na figura 69. Como se pode observar, o clube JIV (Juventude Ilha Verde) demonstra ser o mais popular quer recentemente, quer em termos históricos.

Figura 69 – Distribuição dos 10 Clubes Mais Populares da Atualidade

Na figura 70 é possível observar a quantidade total de atletas distintos por semana nos distritos analisados. A figura demonstra que os meses de verão são os de menor atividade desportiva.

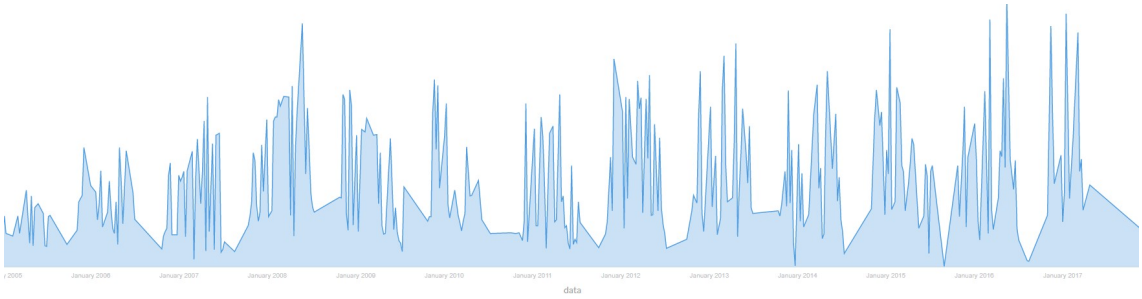


Figura 70 – Evolução da Quantidade de Atletas Participantes Por Semana

Existem muitas variantes, que podem influenciar o rendimento de um atleta, para além dos fatores ambientais (como por exemplo o escalão e género do atleta, o seu treino ao longo do tempo, peso do objeto de lançamento, distância de uma prova, etc.). Por esse motivo, optou-se por verificar apenas um atleta e uma modalidade de cada vez e ao longo do tempo, esperando dessa forma isolar a informação pretendida. Para isso, fez-se uma análise para obter quais os atletas que efetuaram um maior número de provas válidas de lançamento do dardo, arremesso do peso, corrida de 10Km e salto em comprimento (ver figuras 71, 72, 73 e 74).

Atletas que Efetuaram Maior Nº de Lançamentos do Dardo

nome do atleta	Count
Ruben Ventura	144
Elisabete Silva	78
Ricardo Oliveira	77
Mário Florença	62
Renato Cordeiro	62
José Oliveira	54

Rows 1-6 of 1812

Figura 71 – Atletas que Registaram Maior Número de Lançamentos do Dardo

Atletas que Efetuaram Maior Nº de Lançamentos do Peso

nome do atleta	Count
Elisa Pacheco	96
Luís Raposo	69
João Raposo	64
José Oliveira	64
Renato Cordeiro	52
Sara Oliveira	52

Rows 1-6 of 3238

Figura 72 – Atletas que Registaram Maior Número de Arremessos do Peso

Atletas que Efetuaram Maior Nº de Provas de 10 Km

nome do atleta	Count
Marco Amaral	23
Bianca Amaral	21
Vera Amarelo	17
Frederico Medeiros	15
Pedro Amaral	15
Elson Caçador	13

Rows 1-6 of 158

Figura 73 – Atletas que Registaram Maior Número de Corridas de 10Km

Atletas que Efetuaram Maior Nº de Saltos em Comprimento

nome do atleta	Count
Tomás Militão	127
Pedro Amaral	114
João Raposo	92
Ana Câmara	81
Elisa Pacheco	73
Ricardo Oliveira	70

Rows 1-6 of 4352

Figura 74 – Atletas que Registaram Maior Número de Saltos em Comprimento

Estas provas foram selecionadas devido à sua disparidade no que diz respeito ao tipo de esforço físico exigido de forma a observar a influência de fatores ambientais em situações diversas. Os atletas com mais registos (na sua respetiva modalidade), foram selecionados para serem estudadas as suas marcas face às diferentes altitudes e ao longo do tempo, e o resultado obtido pode ser visto na figura 75 para o lançamento do dardo, figura 76 para o arremesso do peso, figura 77 para a corrida de 10Km e figura 78 para o salto em comprimento.

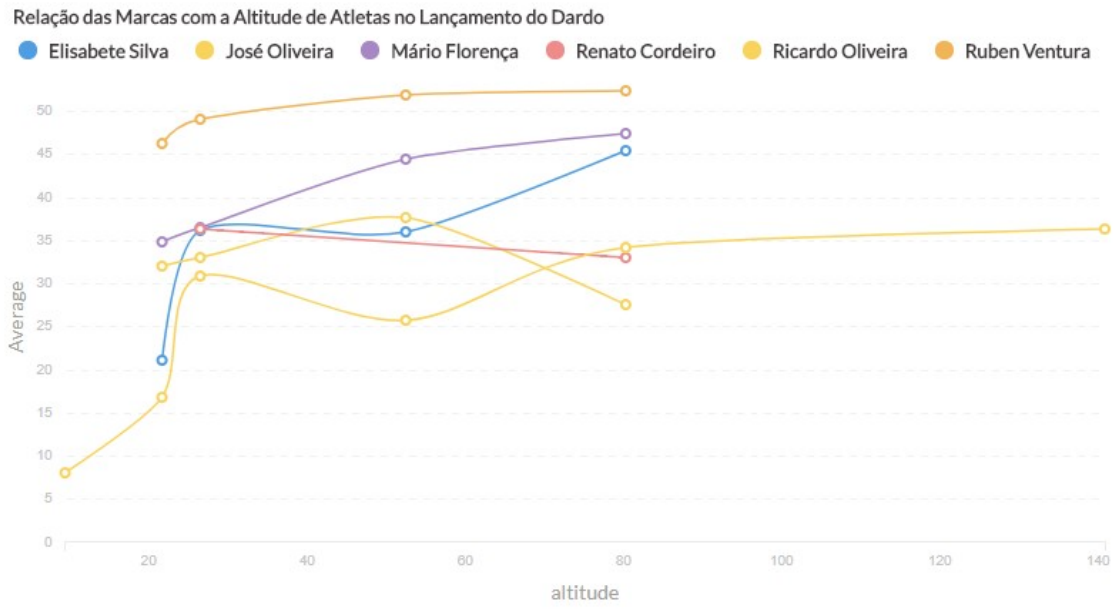


Figura 75 – Média das Marcas Por Altitude de Vários Atletas (Lançamento do Dardo)

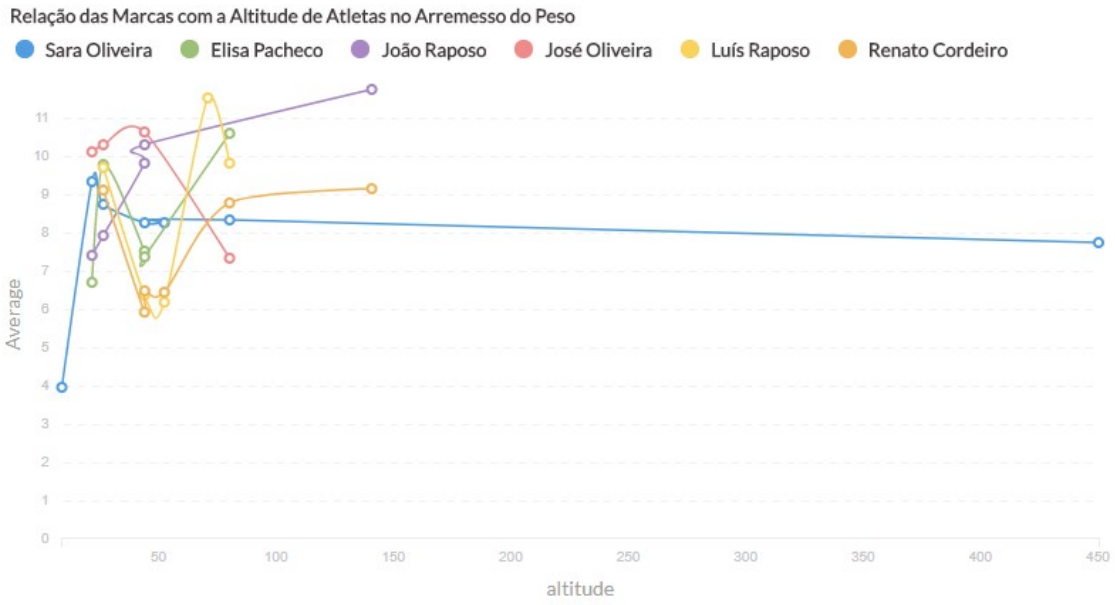


Figura 76 – Média das Marcas Por Altitude de Vários Atletas (Arremesso do Peso)

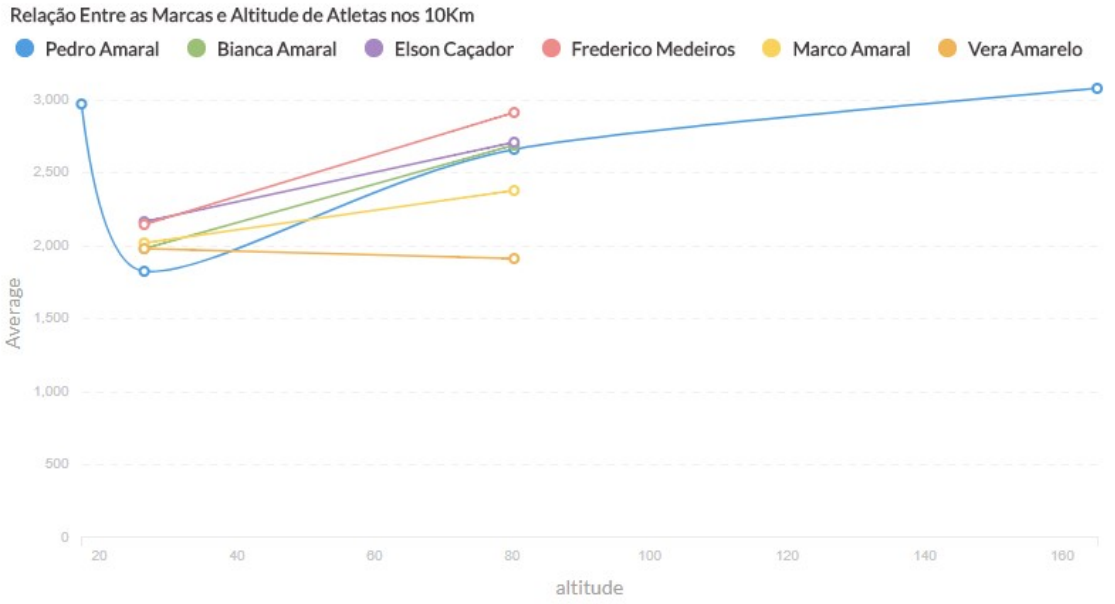


Figura 77 – Média das Marcas Por Altitude de Vários Atletas (Corrida de 10Km)

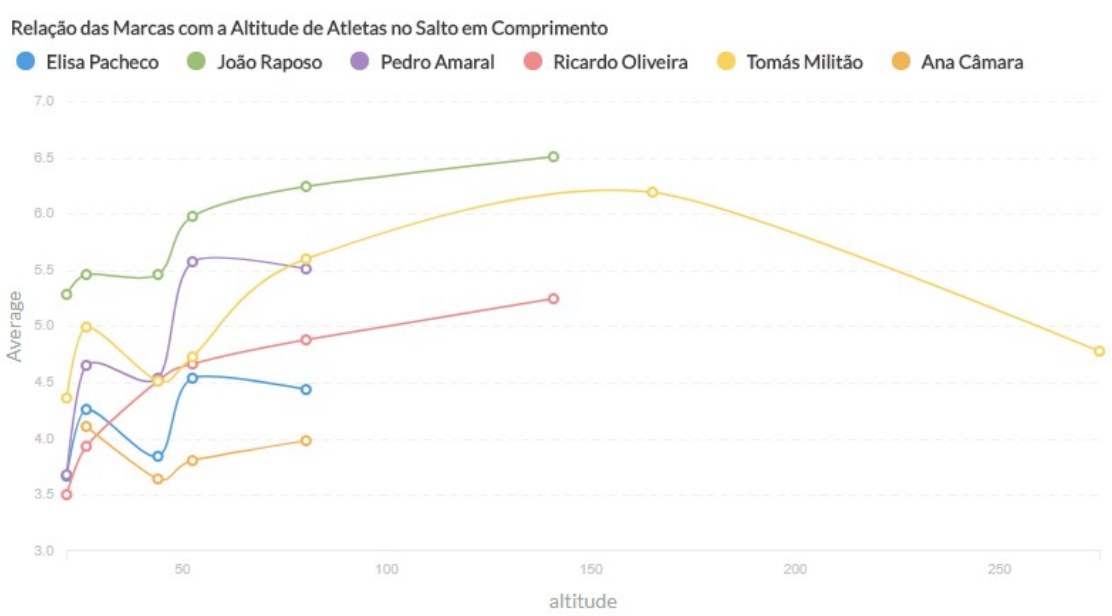


Figura 78 – Média das Marcas Por Altitude de Vários Atletas (Salto em Comprimento)

Com base nas figuras 75, 76, 77 e 78, pode-se afirmar que, na modalidade de corrida de 10Km, em geral, as marcas pioram com o aumento da altitude ao passo que as restantes modalidades parecem piorar.

Para além da altitude, também foi verificada a influência que um dia de chuva pode ter no rendimento dos atletas. O resultado obtido pode ser observado na figura 79 para o lançamento do dardo, figura 80 para o arremesso do peso, figura 81 para a corrida de 10Km e figura 82 para o salto em comprimento.

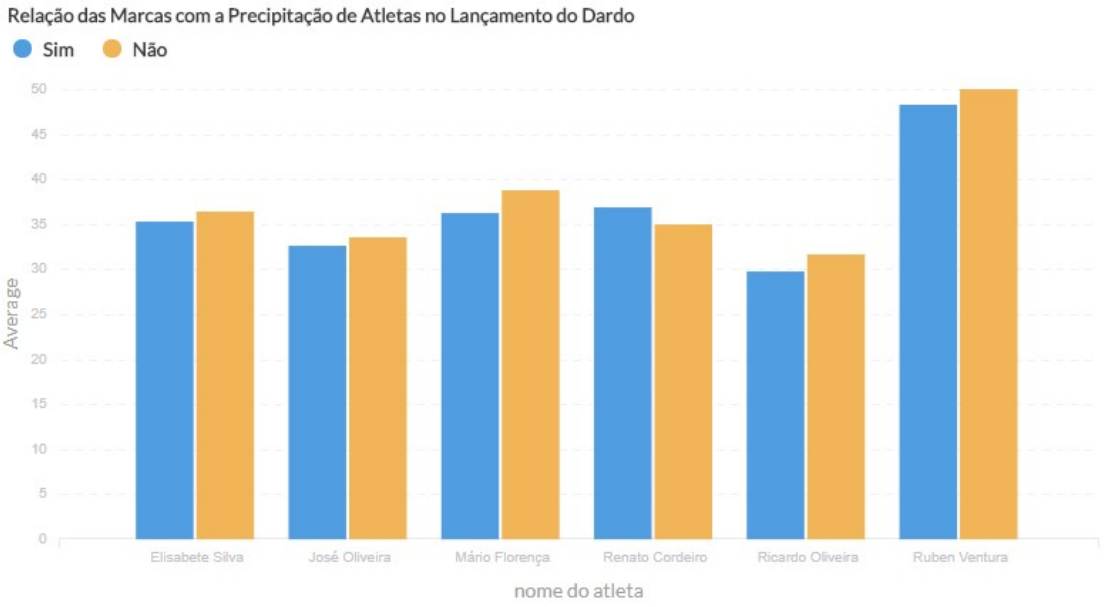


Figura 79 – Média das Marcas Com e Sem Chuva de Vários Atletas (Lançamento do Dardo)

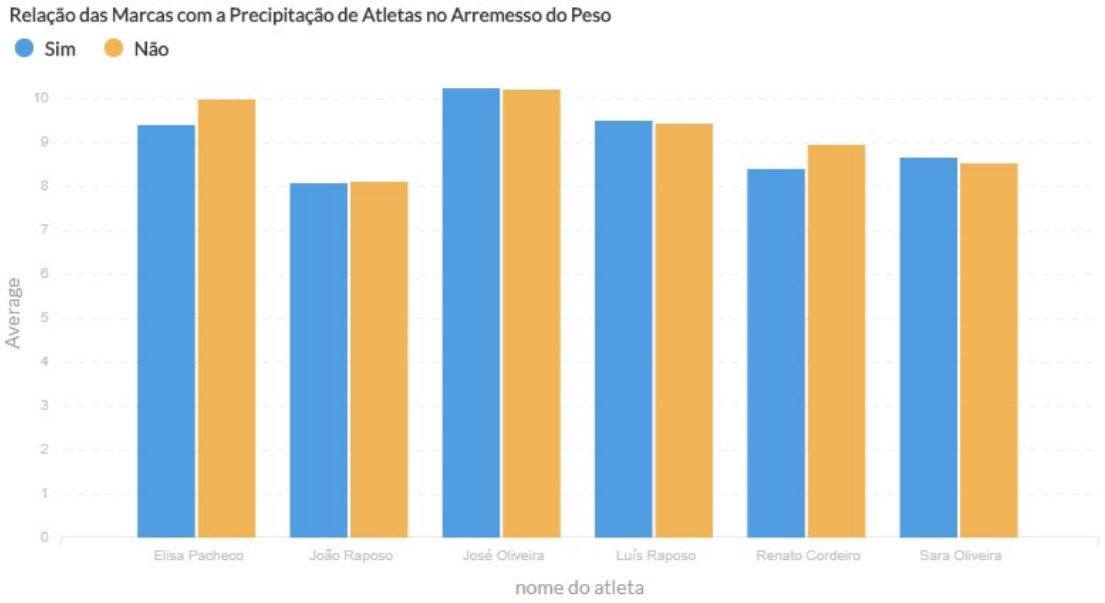


Figura 80 – Média das Marcas Com e Sem Chuva de Vários Atletas (Arremesso do Peso)

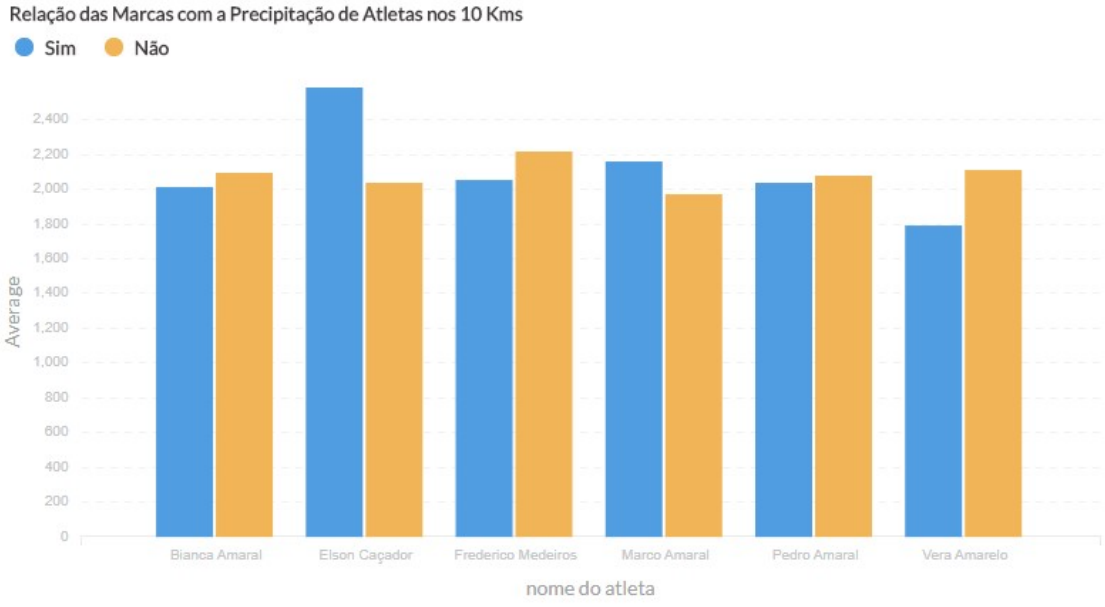


Figura 81 – Média das Marcas Com e Sem Chuva de Vários Atletas (Corrida de 10Km)

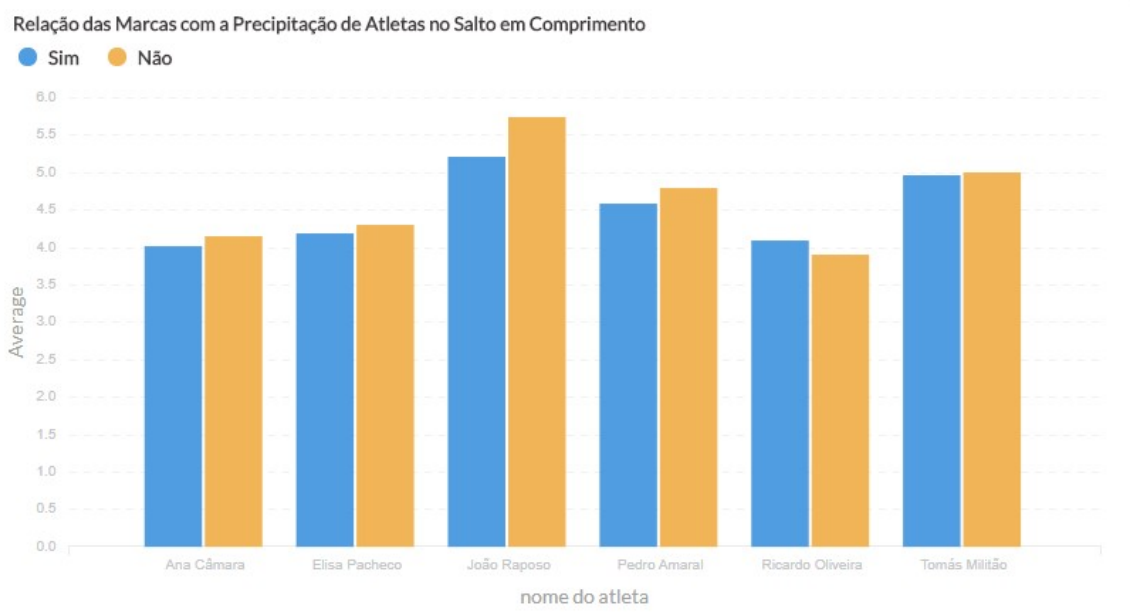


Figura 82 – Média das Marcas Com e Sem Chuva de Vários Atletas (Salto em Comprimento)

As figuras 79, 80, 81 e 82 parecem demonstrar que um dia de chuva parece ter uma influência negativa nas provas de atletismo, com exceção dos 10Kms onde a maior parte dos atletas parecem melhorar ligeiramente.

O "Metabase" também é útil para acompanhar a progressão de um atleta em particular, como por exemplo verificar qual o atleta com mais registos e qual a modalidade mais praticada desse mesmo atleta (ver figura 83).

Atletas com Mais Registos

nome do atleta	Count
Ruben Ventura	596
João Raposo	588
Pedro Amaral	530
Tomás Militão	523
Ricardo Oliveira	465
Marco Amaral	452
Elisa Pacheco	438
André Garcia	435
José Oliveira	370
Miguel Machado	369
Bianca Amaral	361
Ana Câmara	350
Mário Florença	349
Elisabete Silva	344

Rows 1-14 of 10000

Modalidades Mais Praticadas por Ruben Ventura

nome da modalidade	Count
Lançamento do Dardo	152
Salto em Altura	89
Triplo Salto	75
Salto à Vara	63
Corrida	53
Salto em Comprimento	38
Corrida Barreiras	28
Lançamento do Peso	24
Corrida Plana	11
Lançamento do Disco	9
Marcha	4

Figura 83 – Lista de Atleta Com Mais Registos e as Suas Modalidades Mais Praticadas

Também é possível verificar todos os locais onde o atleta com maior número de registos (Ruben Ventura) realizou as suas provas (ver figura 84).

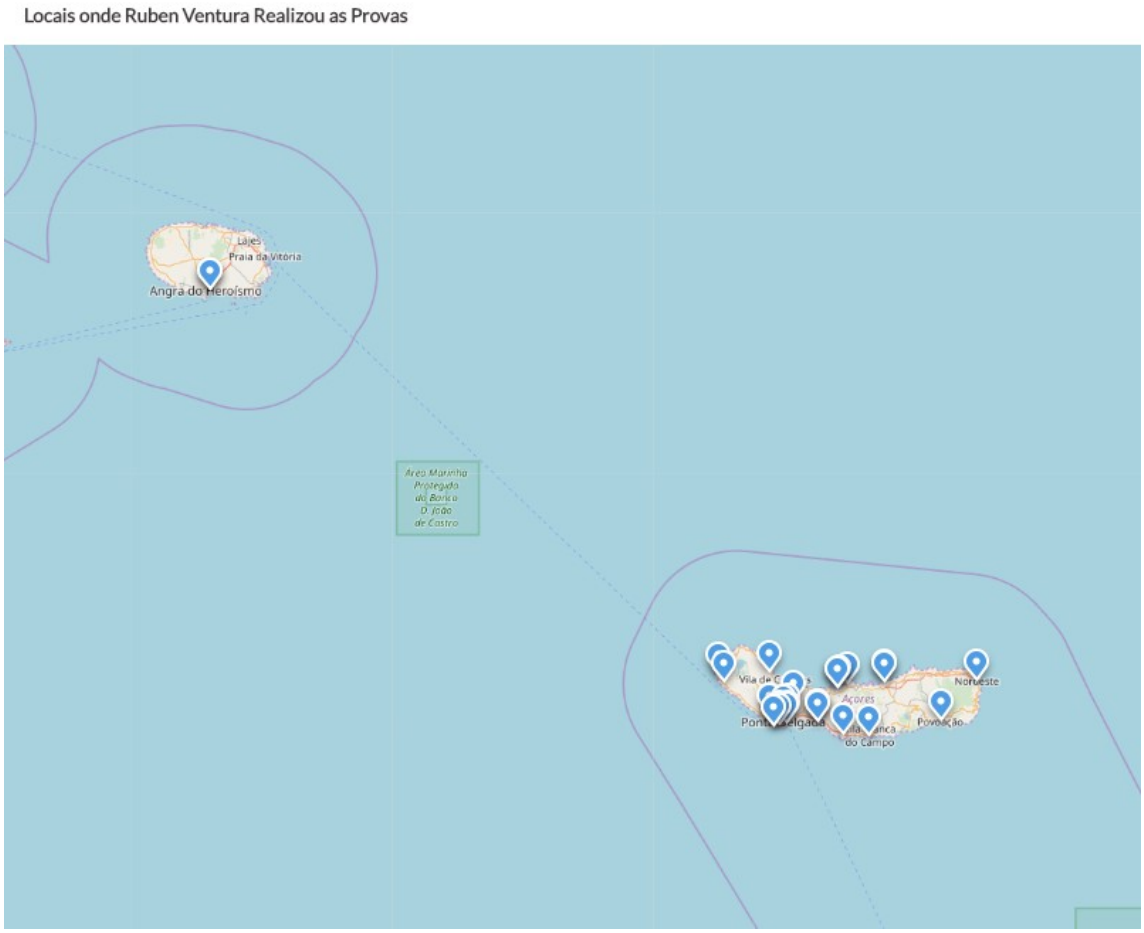


Figura 84 – Locais Onde Ruben Ventura Realizou Provas de Atletismo

Na figura 85 é observado um gráfico com a evolução do atleta Ruben Ventura na modalidade de lançamento do dardo desde 2005 até 2017.

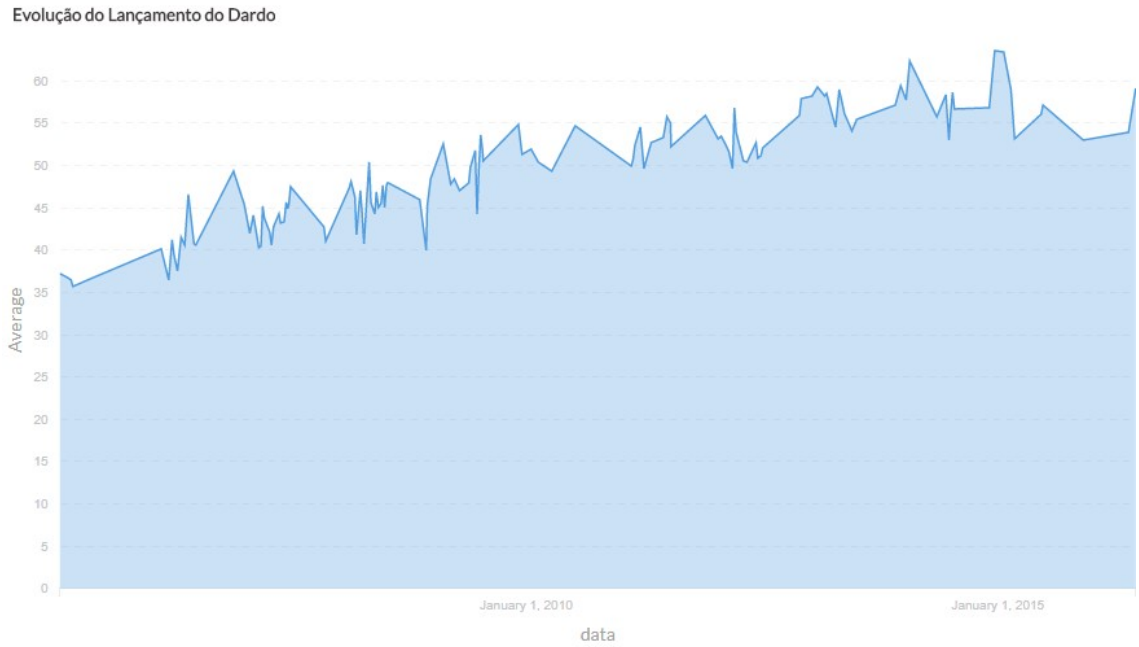


Figura 85 – Marca de Ruben Ventura ao Longo do Tempo (Lançamento do Dardo)

Na figura 86 pode-se observar todos os locais onde Ruben Ventura realizou o lançamento do dardo contra todos os locais onde foram registadas provas (ver figura 87). De notar que de momento apenas os dados dos distritos dos Açores, Aveiro, Beja, Bragança, Vila Real e Viseu foram completamente analisados.

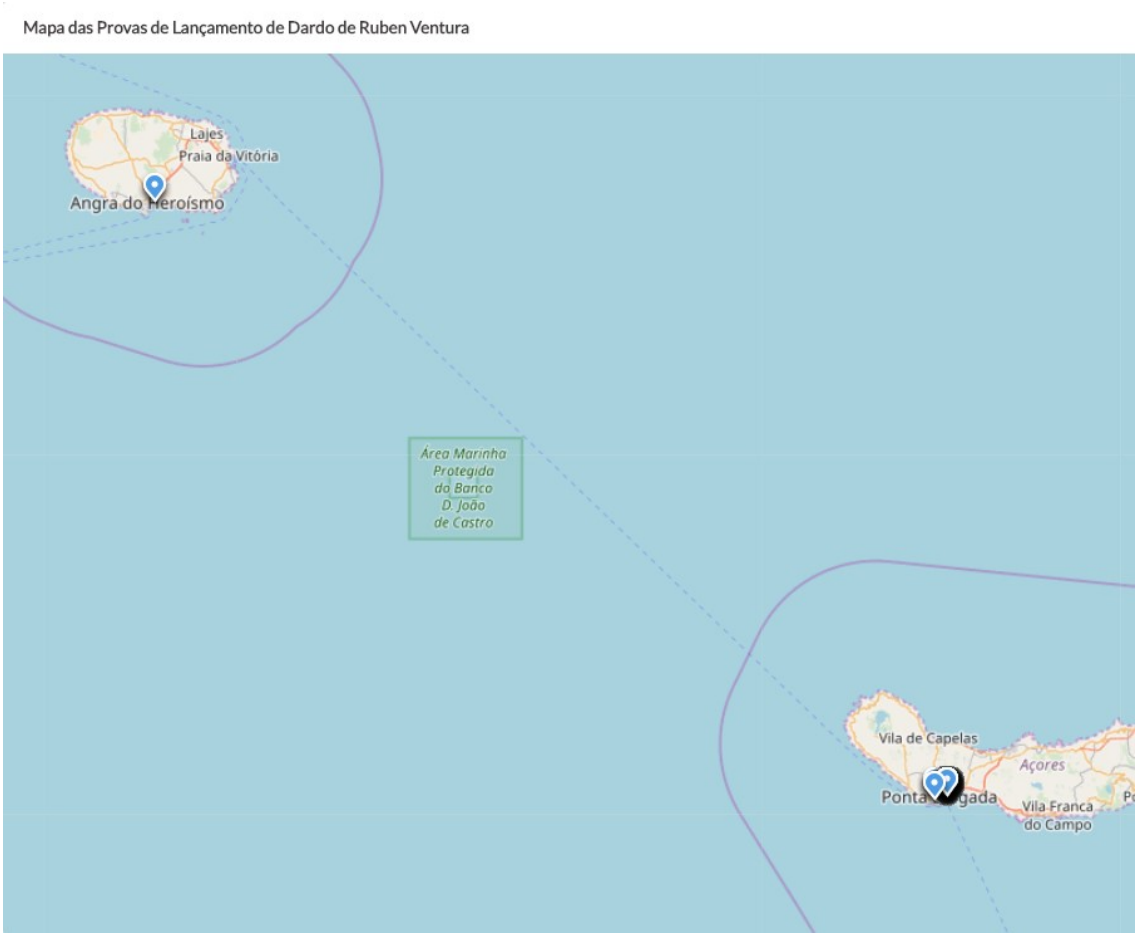


Figura 86 – Locais Onde Ruben Ventura Lançou o Dardo

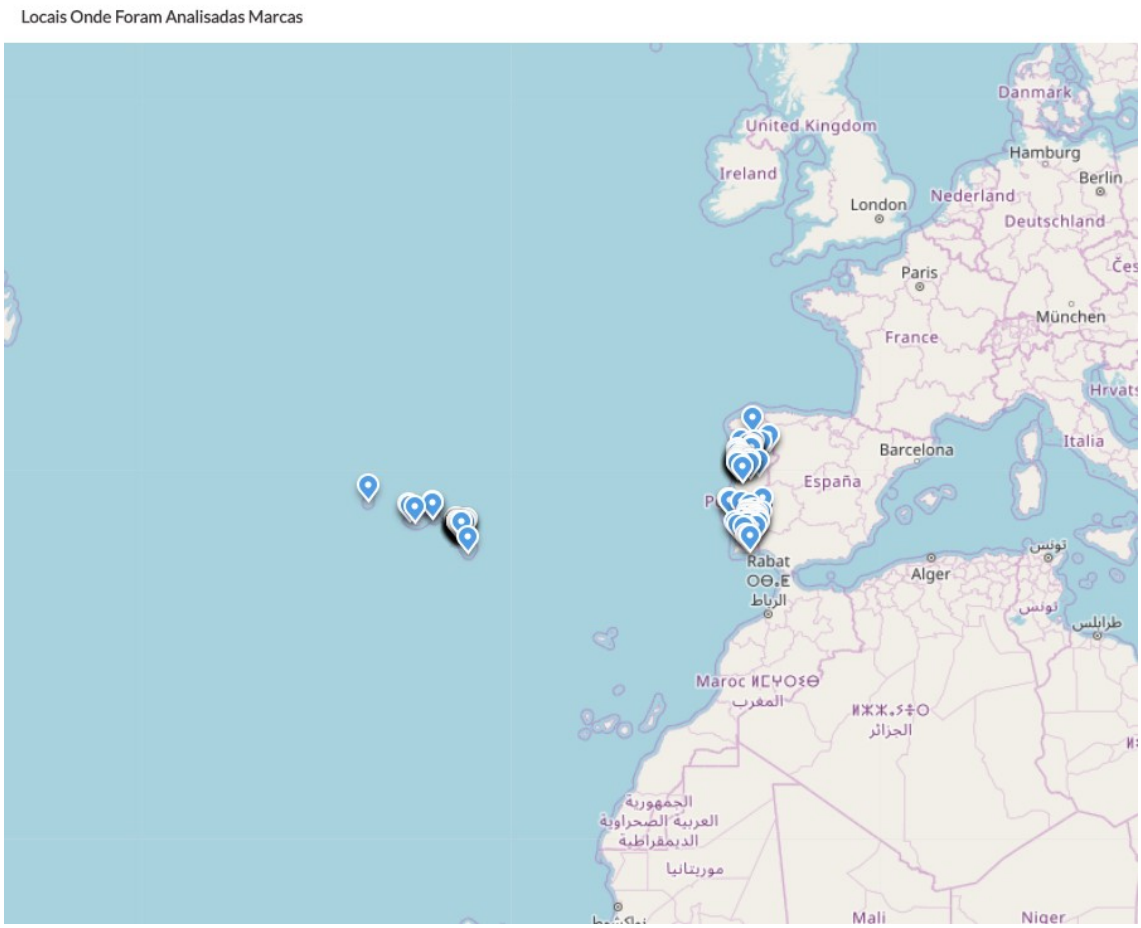


Figura 87 – Locais de Todas as Provas Completamente Analisadas Até Agora

5.3 FASE 8 – DM

Para este projeto, é inicialmente feita uma abordagem indireta com uma análise exploratória dos dados, seguida de uma abordagem direta onde são testados 2 algoritmos, e efetuadas previsões, para um caso em particular.

Primeiro são verificados quais são os atributos mais relevantes para cada tipo de modalidade, para prever uma marca. Para isso, utilizou-se o algoritmo "reliefF" do *widget* "Rank". Os resultados podem ser verificados nas figuras 88 até 100.

	#	RReliefF
atleta	...	0.768
data	...	0.727
prova	...	0.673
clube	...	0.450
direcao_vento	...	0.420
escalao	6	0.339
dia_semana	6	0.231
sexo	2	0.212
dia	C	0.174
humidade	C	0.155

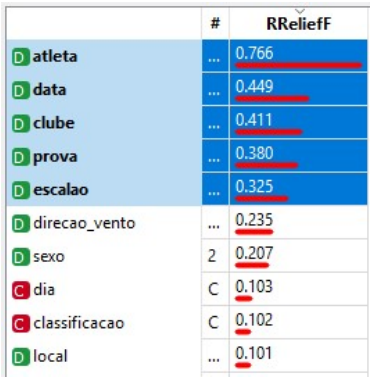
Figura 88 – Ranking do Triplo Salto (Para Marcas)

	#	RReliefF
data	...	0.779
prova	...	0.748
atleta	...	0.640
direcao_vento	...	0.637
clube	...	0.476
escalao	7	0.298
dia	C	0.236
classificacao	C	0.223
dia_semana	4	0.201
velocidade_vento	C	0.196

Figura 89 – Ranking do Salto à Vara (Para Marcas)

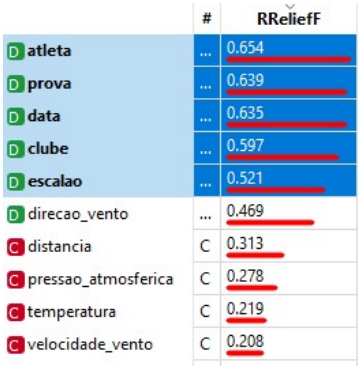
	#	RReliefF
atleta	...	0.796
clube	...	0.563
escalao	...	0.395
sexo	2	0.277
data	...	0.210
prova	...	0.183
direcao_vento	...	0.118
classificacao	C	0.105
dia_semana	7	0.085
serie	...	0.076

Figura 90 – Ranking do Salto em Comprimento (Para Marcas)



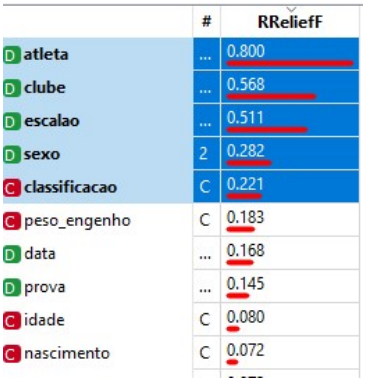
	#	RReliefF
atleta	...	0.766
data	...	0.449
clube	...	0.411
prova	...	0.380
escalao	...	0.325
direcao_vento	...	0.235
sexo	2	0.207
dia	C	0.103
classificacao	C	0.102
local	...	0.101

Figura 91 – Ranking do Salto em Altura (Para Marcas)



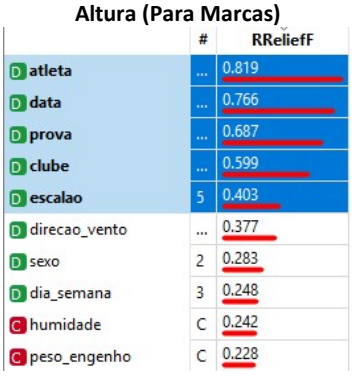
	#	RReliefF
atleta	...	0.654
prova	...	0.639
data	...	0.635
clube	...	0.597
escalao	...	0.521
direcao_vento	...	0.469
distancia	C	0.313
pressao_atmosferica	C	0.278
temperatura	C	0.219
velocidade_vento	C	0.208

Figura 92 – Ranking da Marcha (Para Marcas)



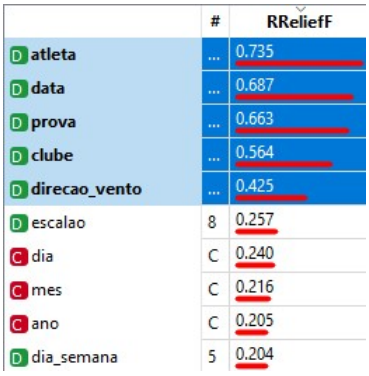
	#	RReliefF
atleta	...	0.800
clube	...	0.568
escalao	...	0.511
sexo	2	0.282
classificacao	C	0.221
peso_engenho	C	0.183
data	...	0.168
prova	...	0.145
idade	C	0.080
nascimento	C	0.072

Figura 93 – Ranking do Arremesso do Peso (Para Marcas)



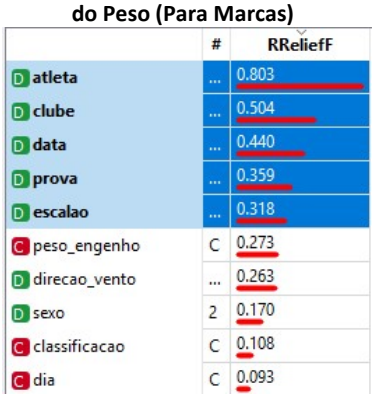
	#	RReliefF
atleta	...	0.819
data	...	0.766
prova	...	0.687
clube	...	0.599
escalao	5	0.403
direcao_vento	...	0.377
sexo	2	0.283
dia_semana	3	0.248
humidade	C	0.242
peso_engenho	C	0.228

Figura 94 – Ranking do Lançamento do Martelo (Para Marcas)



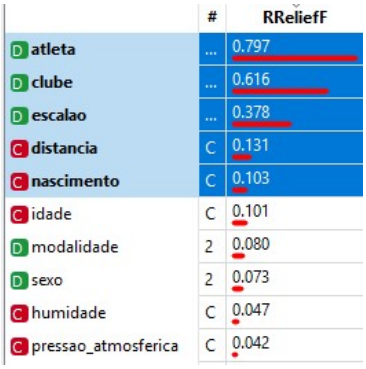
	#	RReliefF
atleta	...	0.735
data	...	0.687
prova	...	0.663
clube	...	0.564
direcao_vento	...	0.425
escalao	8	0.257
dia	C	0.240
mes	C	0.216
ano	C	0.205
dia_semana	5	0.204

Figura 95 – Ranking do Lançamento do Disco (Para Marcas)



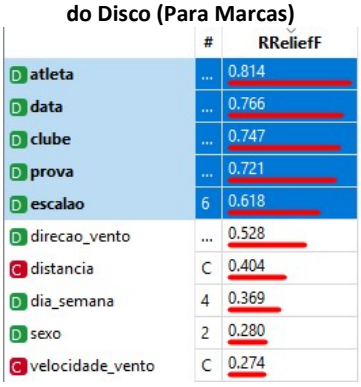
	#	RReliefF
atleta	...	0.803
clube	...	0.504
data	...	0.440
prova	...	0.359
escalao	...	0.318
peso_engenho	C	0.273
direcao_vento	...	0.263
sexo	2	0.170
classificacao	C	0.108
dia	C	0.093

Figura 96 – Ranking do Lançamento do Dardo (Para Marcas)



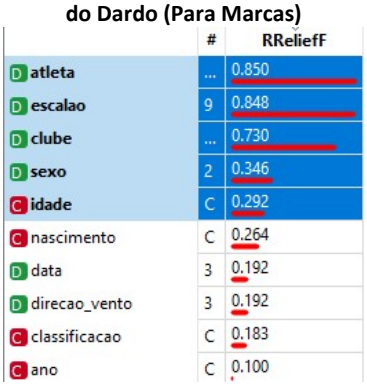
	#	RReliefF
atleta	...	0.797
clube	...	0.616
escalao	...	0.378
distancia	C	0.131
nascimento	C	0.103
idade	C	0.101
modalidade	2	0.080
sexo	2	0.073
humidade	C	0.047
pressao_atmosferica	C	0.042

Figura 97 – Ranking da Corrida de Pista, Estrada e Trail (Para Marcas)



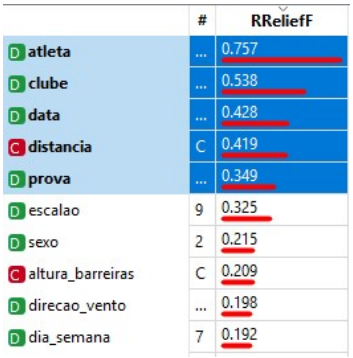
	#	RReliefF
atleta	...	0.814
data	...	0.766
clube	...	0.747
prova	...	0.721
escalao	6	0.618
direcao_vento	...	0.528
distancia	C	0.404
dia_semana	4	0.369
sexo	2	0.280
velocidade_vento	C	0.274

Figura 98 – Ranking da Corrida de Obstáculos (Para Marcas)



	#	RReliefF
atleta	...	0.850
escalao	9	0.848
clube	...	0.730
sexo	2	0.346
idade	C	0.292
nascimento	C	0.264
data	3	0.192
direcao_vento	3	0.192
classificacao	C	0.183
ano	C	0.100

Figura 99 – Ranking da Corrida de Corta Mato (Para Marcas)



	#	RReliefF
atleta	...	0.757
clube	...	0.538
data	...	0.428
distancia	C	0.419
prova	...	0.349
escalao	9	0.325
sexo	2	0.215
altura_barreiras	C	0.209
direcao_vento	...	0.198
dia_semana	7	0.192

Figura 100 – Ranking da Corrida de Barreiras (Para Marcas)

A mesma avaliação foi feita também para prever a classificação de um atleta, como pode ser visto nas figuras 101 até 113.

	#	RReliefF
atleta	...	0.794
data	...	0.733
prova	...	0.685
clube	...	0.576
direcao_vento	...	0.465
escalao	6	0.294
dia_semana	6	0.229
humidade	C	0.184
sexo	2	0.178
dia	C	0.170

Figura 101 – Ranking do Triplo Salto (Para Classificações)

	#	RReliefF
data	...	0.779
prova	...	0.755
atleta	...	0.731
direcao_vento	...	0.668
clube	...	0.401
escalao	7	0.286
dia	C	0.245
dia_semana	4	0.233
marca	C	0.206
velocidade_vento	C	0.196

Figura 102 – Ranking do Salto à Vara (Para Classificações)

	#	RReliefF
atleta	...	0.812
clube	...	0.521
escalao	...	0.394
sexo	2	0.245
data	...	0.136
prova	...	0.126
direcao_vento	...	0.092
idade	C	0.044
nascimento	C	0.044
dia	C	0.030

Figura 103 – Ranking do Salto em Comprimento (Para Classificações)

	#	RReliefF
atleta	...	0.798
clube	...	0.509
data	...	0.435
prova	...	0.360
escalao	...	0.294
direcao_vento	...	0.279
sexo	2	0.170
dia	C	0.112
pressao_atmosferica	C	0.096
ano	C	0.084

Figura 104 – Ranking do Salto em Altura (Para Classificações)

	#	RReliefF
atleta	...	0.739
data	...	0.526
prova	...	0.463
direcao_vento	...	0.398
clube	...	0.366
marca	C	0.349
escalao	...	0.302
sexo	2	0.206
serie	8	0.198
dia	C	0.147

Figura 105 – Ranking da Marcha (Para Classificações)

	#	RReliefF
atleta	...	0.820
clube	...	0.573
escalao	...	0.452
sexo	2	0.246
data	...	0.148
peso_engenho	C	0.114
prova	...	0.107
direcao_vento	...	0.091
marca	C	0.087
nascimento	C	0.048

Figura 106 – Ranking do Arremesso do Peso (Para Classificações)

	#	RReliefF
atleta	...	0.801
data	...	0.736
clube	...	0.653
prova	...	0.619
direcao_vento	...	0.410
escalao	5	0.335
dia_semana	3	0.291
peso_engenho	C	0.268
sexo	2	0.251
ano	C	0.240

Figura 107 – Ranking do Lançamento do Martelo (Para Classificações)

	#	RReliefF
atleta	...	0.817
clube	...	0.670
data	...	0.644
prova	...	0.569
direcao_vento	...	0.381
escalao	8	0.238
dia_semana	5	0.228
sexo	2	0.199
dia	C	0.178
peso_engenho	C	0.167

Figura 108 – Ranking do Lançamento do Disco (Para Classificações)

	#	RReliefF
atleta	...	0.776
data	...	0.510
clube	...	0.504
prova	...	0.419
direcao_vento	...	0.332
escalao	...	0.322
marca	C	0.181
peso_engenho	C	0.176
dia_semana	6	0.134
dia	C	0.124

Figura 109 – Ranking do Lançamento do Dardo (Para Classificações)

	#	RReliefF
atleta	...	0.798
clube	...	0.555
escalao	...	0.372
sexo	2	0.134
nascimento	C	0.083
idade	C	0.078
serie	...	0.069
modalidade	2	0.038
marca	C	0.031
visibilidade	C	0.023

Figura 110 – Ranking da Corrida de Pista, Estrada ou Trail (Para Classificações)

	#	RReliefF
atleta	...	0.821
data	...	0.735
prova	...	0.664
clube	...	0.648
direcao_vento	...	0.504
escalao	6	0.376
dia_semana	4	0.330
visibilidade	C	0.277
humidade	C	0.268
velocidade_vento	C	0.263

Figura 111 – Ranking da Corrida de Obstáculos (Para Classificações)

	#	RReliefF
atleta	...	0.838
clube	...	0.740
escalao	9	0.629
sexo	2	0.363
marca	C	0.224
idade	C	0.134
nascimento	C	0.133
data	3	0.118
direcao_vento	3	0.118
humidade	C	0.062

Figura 112 – Ranking da Corrida de Corta Mato (Para Classificações)

	#	RReliefF
atleta	...	0.803
clube	...	0.516
escalao	9	0.384
data	...	0.338
prova	...	0.316
sexo	2	0.240
direcao_vento	...	0.224
altura_barreiras	C	0.155
dia	C	0.107
serie	...	0.092

Figura 113 – Ranking da Corrida de Barreiras (Para Classificações)

As figuras demonstram que os atletas são o atributo que melhor define tanto o resultado das marcas como das classificações para quase todos os tipos de modalidade. Tomando como exemplo a modalidade do lançamento do martelo é possível observar que as distribuições, das marcas de cada atleta, são na sua maioria bastante distintas umas das outras (figura 114), confirmando a observação anterior.

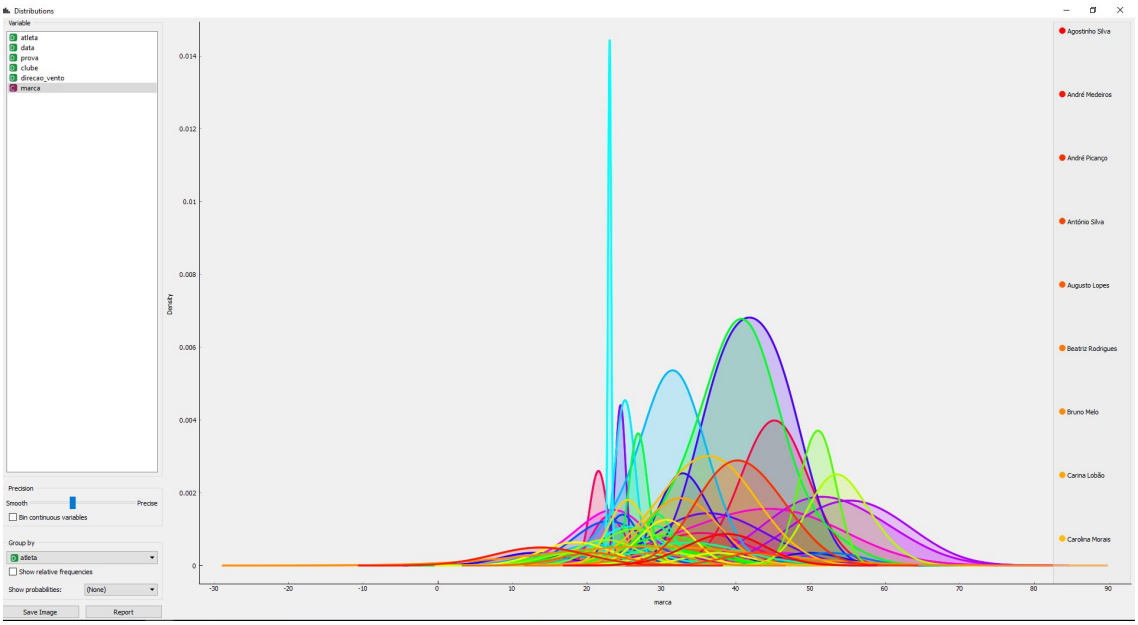


Figura 114 – Distribuição das Marcas do Lançamento do Martelo de Cada Atleta

Um estudo sobre quais os atributos meteorológicos que mais influenciam no lançamento do dardo de Ruben Ventura também foi feito. Para obter um maior sucesso na previsão o atributo alvo “marca” foi discreteado em 10 intervalos, tal como observado na figura 115.

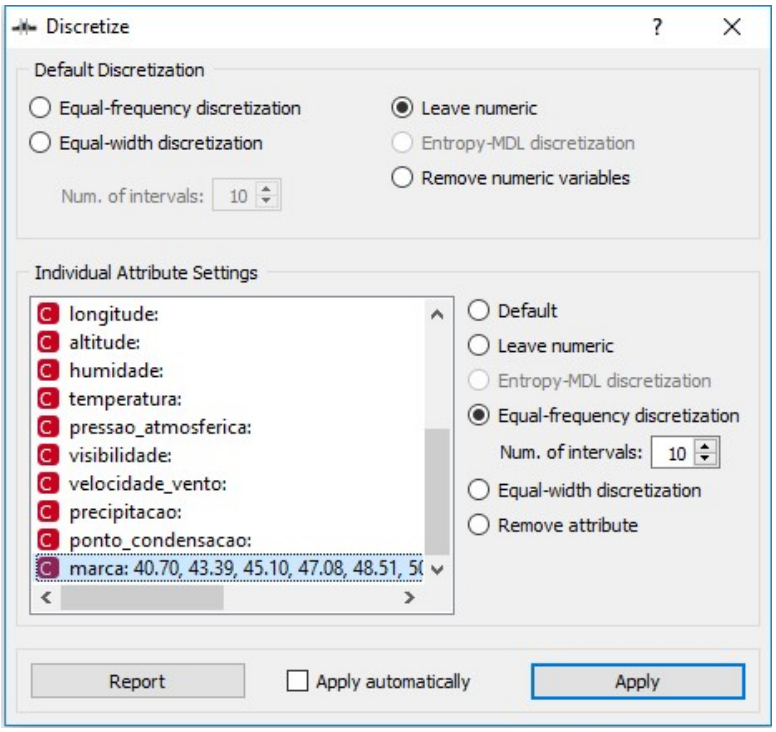


Figura 115 – Discretização da Marca

Seguidamente verificou-se qual a combinação de 2 atributos que melhor caracterizam uma marca. Na figura 116 é observado que a pressão atmosférica com a temperatura é a combinação de maior utilidade para classificar as marcas de atletismo de Ruben Ventura no lançamento do dardo (tendo em conta os dados existente até ao momento).

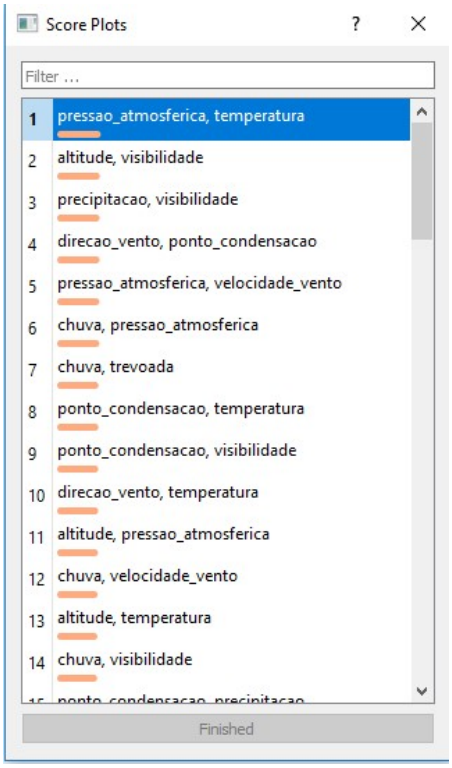


Figura 116 – Ranking de 2 Atributos num Determinado Contexto

Dois algoritmos foram testados para verificar qual o melhor a prever as marcas no contexto explicado anteriormente. Os algoritmos testados foram a “floresta aleatória” e “kNN”. Na figura 117 é possível observar que a “floresta aleatória ” é a solução mais apta para este cenário. Múltiplas classificações foram usadas para avaliar os algoritmos, de entre as quais, a exatidão (CA), precisão (Precision), sensibilidade (Recall) e uma média da sensibilidade e precisão (F1).

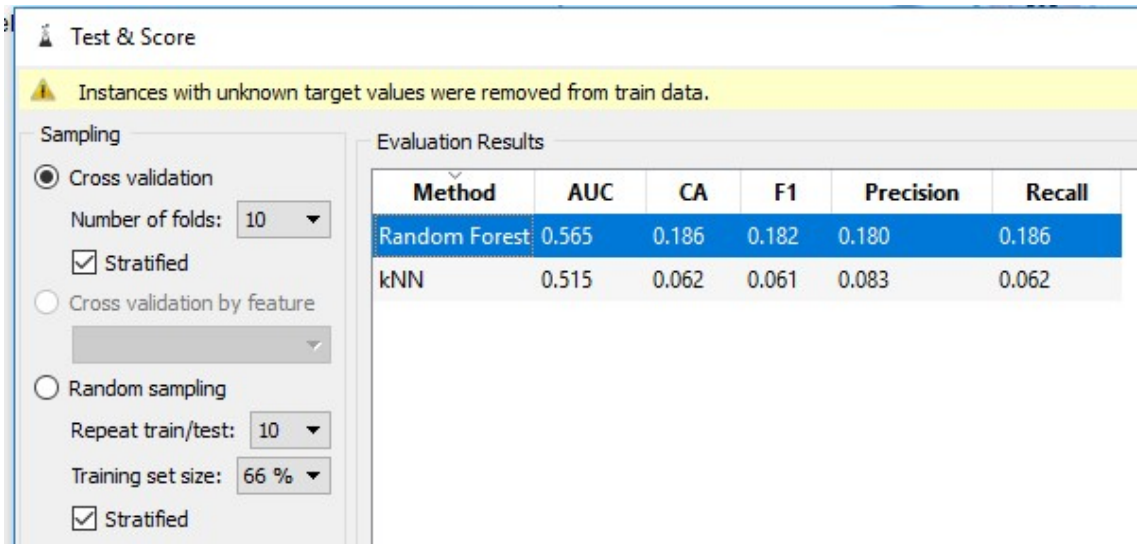


Figura 117 – Avaliação dos Algoritmos de DM num Determinado Contexto

Como confirmação dos resultados dos testes estão abaixo as figuras 118 e 119 para as matrizes da confusão da “floresta aleatória” e “kNN”. Nelas é possível observar a quantidade de previsões corretas e incorretas de cada estado da variável a prever (marcas dos atletas).

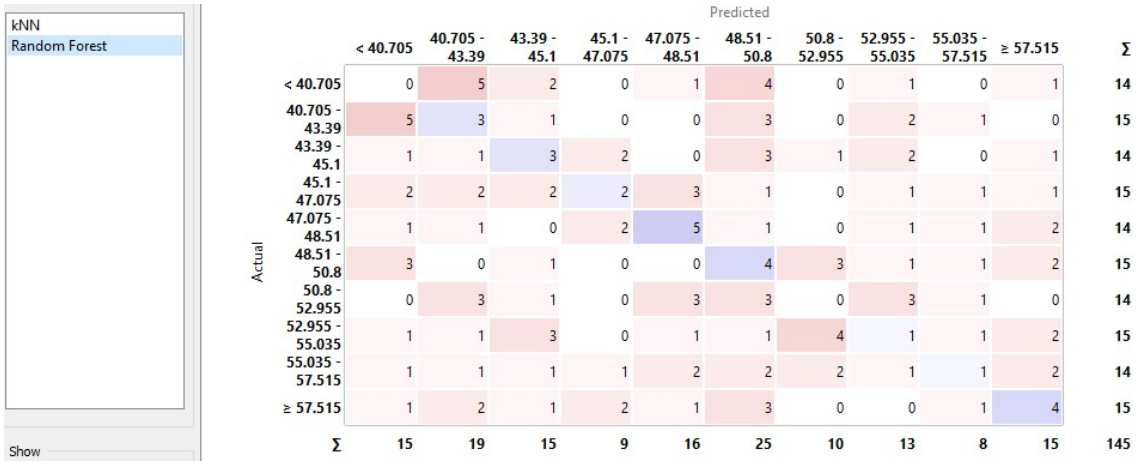


Figura 118 – Matriz da Confusão Para Floresta Aleatória

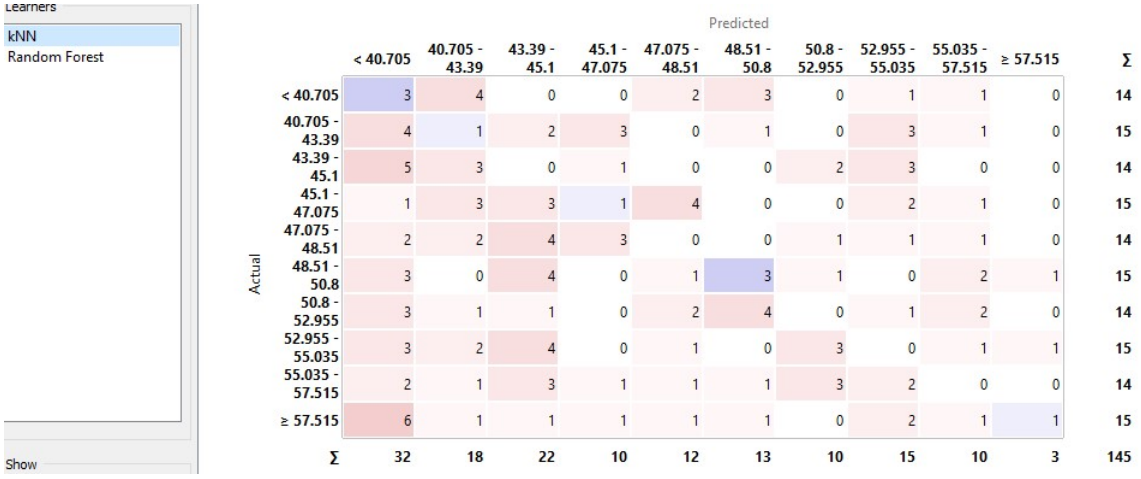


Figura 119 – Matriz da Confusão Para kNN

Utilizou-se o algoritmo que provou ser mais fiável para prever futuras marcas de lançamento do dardo de Ruben Ventura (tendo por base apenas fatores meteorológicos). Para todos os casos, todos os fatores meteorológicos são constantes, com exceção da temperatura:

- Altitude – 26.52m;
- Humidade – 67%;
- Pressão atmosférica – 1029.02hPa;
- Visibilidade – 12.6Km;
- Velocidade do vento – 9Km/h;
- Direção do vento – Variável;
- Precipitação – 1mm;
- Nevoeiro – Não;
- Chuva – Não;
- Trovoada – Não;
- Ponto de condensação – 18°C.

Com se pode observar na figura 120, no primeiro caso, para uma temperatura de 17°C, existe uma probabilidade de 21% que a próxima marca esteja abaixo dos 40.705 metros. No segundo caso para uma temperatura de 18°C é mais provável que o atleta realize um lançamento entre os 55.035 e os 57.515, com uma percentagem de 28% e se a temperatura subir para 20°C essa percentagem sobe para os 30%.

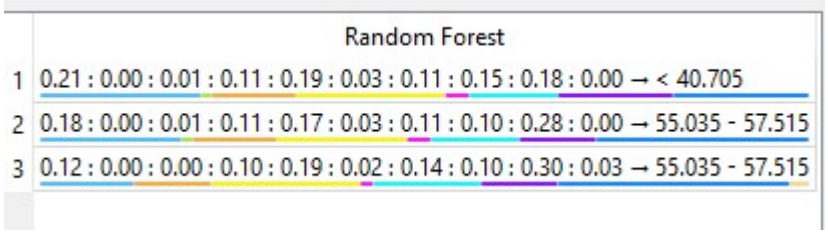


Figura 120 – Previsões de Futuros Lançamentos de Ruben Ventura

6. CONCLUSÃO

Com este projeto criou-se, através de um processo semiautomático, uma base de dados de atletismo português, com resultados de provas realizadas a nível distrital. Estes dados, recolhidos de sítios Web das várias distritais do país, são depois integrados com os dados sobre localização, altitude e condições atmosféricas em que foram realizadas as provas.

Neste capítulo serão referidas, com maior detalhe, as conclusões retiradas durante o tratamento dos dados, o processo de análise dos dados (BI e DM) e traçadas as linhas gerais daquilo que poderá ser o trabalho futuro.

6.1 TRATAMENTO DOS DADOS

Cerca de 90% dos ficheiros disponibilizados on-line com resultados oficiais de atletismo em Portugal, são apresentados no formato PDF.

Ao contrário de páginas HTML (HyperText Markup Language), o formato PDF dificulta qualquer trabalho de análise de dados, pois como já referido no capítulo 2.9 os ficheiros PDF podem-se rapidamente tornar em "armadilhas". Cada ficheiro processado pode apresentar especificidades diferentes. É necessário, ao longo do tempo, identificar padrões e tê-los em consideração programaticamente, para além de verificar se o código efetua uma importação precisa (Kazil, J. & Jarmul, K., 2016).

Por outro lado, não devemos analisar algo apenas porque existe em grande quantidade ou é de fácil acesso. É muito comum este tipo de abordagem terminar em tempo perdido e sem resultados (McCallum, Q., 2013).

O protótipo desenvolvido no contexto deste projeto (*PositionParser*) provou ser capaz de obter dados de grande utilidade que, de outra forma, seriam impossíveis de obter rapidamente.

6.2 ANÁLISE DOS DADOS

Até ao momento, foram analisados 6 distritos, Açores, Aveiro, Beja, Bragança, Vila Real e Viseu, que representam cerca de 14% da informação já analisada (1323 ficheiros ou 166277 registos), mas não limpa (demonstrado na figura 63). Com estes dados foi possível obter conclusões importantes a partir da análise de BI e DM. Todas as conclusões obtidas têm como referência, por enquanto esses distritos.

A ferramenta de BI utilizada (Metabase) atualiza automaticamente todos os gráficos e tabelas, criados pelo utilizador, sempre que nova informação é introduzida. Desta forma a visualização dos dados não fica excessivamente dependente (acoplamento apertado) do momento em que os dados são introduzidos.

Através duma análise geral, de BI, aos dados já tratados, observa-se que o atletismo português é um desporto onde o sexo masculino é mais predominante. Cerca de 6 em cada 10 atletas que registaram marcas são do género masculino (ver figura 64). O

escalão com mais atletas é o de iniciados com mais de 15% (ver figura 65), a modalidade mais praticada é a corrida (ver figura 66). O clube mais popular é o JIV (Juventude Ilha Verde) que tem aumentado a sua popularidade desde 2012 até agora (demonstrado nas figuras 67, 68 e 69).

Na figura 70, verifica-se que todos os anos a quantidade de participações de atletas por semana parece seguir um padrão semelhante onde os meses de Inverso são os de maior atividade.

Através de uma análise de BI verifica-se que uma maior altitude parece melhorar modalidades explosivas (lançamento do dardo, arremesso do peso e salto em comprimento) ao passo que modalidades de resistência (corrida de 10Km) parecem tornar-se mais difíceis. Isto talvez se deva ao facto de que modalidades explosivas estejam mais dependentes da resistência do ar e força de gravidade ao passo que provas de resistência estejam mais dependentes dos níveis de oxigeno que um atleta consome. Um dia de chuva parece piorar a performance de modalidades explosivas, mas oferecer uma vantagem em corridas de resistência. De realçar, no entanto, a performance do atleta Elson Caçador, que parece ser bastante melhor em dias sem chuva (ver figuras de 75 a 82).

Na análise de BI o que mais se destacou foi no acompanhamento individual de cada atleta onde qualquer pessoa pode ter uma visão esclarecedora e intuitiva do progresso de um atleta e locais de provas (demonstrado nas figuras de 84 a 87).

Recorrendo a uma análise exploratória de DM é possível concluir que de todos os atributos existentes, o atleta é o mais determinante tanto para a marca como para a sua classificação, para quase todas as modalidades de atletismo. Isto faz pressupor que, para um clube ou treinador, obter um atleta de grande potencial tem maior prioridade do que fatores ambientais, locais das provas, género, escalão ou a sua progressão ao longo do tempo (ver figuras 88 até 113).

Os resultados do estudo mais específico ao atleta Ruben Ventura (atleta com mais registos) demonstram que, de todos os fatores atmosféricos, a pressão atmosférica com a temperatura é a combinação que mais influencia os seus lançamentos do dardo (modalidade com mais registos de Ruben Ventura), como demonstrado na figura 116.

Utilizando o algoritmo “floresta aleatória” na previsão de futuros lançamentos (ver figura 120) deduz-se que a temperatura ideal de Ruben Ventura no lançamento do dardo situa-se entre os 18 e 20°C.

6.3 Trabalho Futuro

Como trabalho futuro pretende-se completar o DW criado com o carregamento dos dados referentes aos distritos em falta. Além disso, pretende-se fazer a análise de ficheiros referentes a novas épocas de atletismo.

Como trabalho futuro pretende-se também melhorar o protótipo (*PositionParser*) criado durante esta investigação. Para isso, será necessária a criação de uma ferramenta com uma interface gráfica amigável e fácil de usar por qualquer utilizador.
Presume-se que o que torna este protótipo único na análise de tabelas não-estruturadas está assente em 2 fundamentos:

1. Todo o texto é inicialmente dividido em fragmento homogéneos chamados de "tokens", que trazem consistência ao processo de análise (tal como referido no capítulo 4.5);
2. Os "tokens" são relacionados entre si por atributos partilhados por todos (significado, posição relativa e absoluta e padrões de expressão regular).

REFERÊNCIAS

Von Alan, R. Hevner, et al. "Design science in information systems research." *MIS quarterly* 28.1 (2004): 75-105.

Peffers, Ken, et al. "The design science research process: a model for producing and presenting information systems research." *Proceedings of the first international conference on design science research in information systems and technology* (DESRIST 2006). sn, 2006.

Von Alan, R. Hevner, et al. "Design science in information systems research." *MIS quarterly* 28.1 (2004): 75-105.

Vasconcelos, J. B. (2015). Python Algoritmia e Programação Web. Lisboa: FCA.

Mitchell, R. (2015). Web Scraping with Python. Sebastopol, CA: O'Reilly Media, Inc.

Kazil, J. & Jarmul, J. (2016). Data Wrangling with Python. Sebastopol, CA: O'Reilly Media, Inc.

Caldeira, C. P. (2012). Data Warehousing Conceitos e Modelos Com Exemplos Práticos. R. Cidade de Manchester, Lisboa: Edições Sílabo, Lda.

Santos, M. Y. & Ramos, I. (2009). Business Intelligence Tecnologias da Informação na Gestão de Conhecimento. R. D. Estefânia, Lisboa: FCA.

Fayyad, Usama, Gregory Piatetsky-Shapiro, and Padhraic Smyth. "From data mining to knowledge discovery in databases." *AI magazine* 17.3 (1996): 37.

Santos, M. F. & Azevedo, C. (2005). Data Mining Descoberta de Conhecimento em Bases de Dados R. D. Estefânia, Lisboa: FCA.

McCallum, Q. (2013). Bad Data. Sebastopol, CA: O'Reilly Media, Inc.

Perez-Arriaga, Martha O., Trilce Estrada, and Soraya Abad-Mota. "TAO: System for Table Detection and Extraction from PDF Documents." *The Twenty-Ninth International Flairs Conference*. 2016.

Dadachev, Boris, Alexander Balinsky, and Helen Balinsky. "On automatic text segmentation." *Proceedings of the 2014 ACM symposium on Document engineering*. ACM, 2014.

Reitz, K. & Schlusser, T.(2016). The Hitchhiker's Guide to Python.Sebastopol, CA: O'Reilly Media, Inc.

Demšar, Janez. "Data Mining". *Material for Lectures on Data Mining at Kyoto University, Dept. of Health Informatics*. 2010.

Kandel, Sean, et al. "Wrangler: Interactive visual specification of data transformation scripts." *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2011

Khusro, Shah, Asima Latif, and Irfan Ullah. "*On methods and tools of table detection, extraction and annotation in PDF documents.*" Journal of Information Science 41.1 (2015): 41-57.

Bienz, Tim, Richard Cohn, and Adobe Systems (Mountain View, Calif.). *Portable document format reference manual*. Reading, MA, USA: Addison-Wesley, 1993.

Yildiz, Burcu, Katharina Kaiser, and Silvia Miksch. "*pdf2table: A method to extract table information from pdf files.*" IICAI. 2005.

Oro, Ermelinda, and Massimo Ruffolo. "*TREX: An approach for recognizing and extracting tables from PDF documents.*" Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on. IEEE, 2009.

Pitale, Sarang, and Tripti Sharma. "*Information extraction tools for portable document format.*" (2011).

Hassan, Tamir, and Robert Baumgartner. "*Table recognition and understanding from pdf files.*" Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on. Vol. 2. IEEE, 2007.